AUCIL: An Inclusion List Design for Rational Parties

Sarisht Wadhwa*§, Julian Ma†, Thomas Thiery†, Barnabe Monnot†, Luca Zanolini†, Fan Zhang‡ and Kartik Nayak*
*Duke University
†Ethereum Research
‡Yale University
§Email ID: sarisht.wadhwa@duke.edu

***Abstract*—The decentralized nature of blockchains is touted to provide censorship resistance. However, in reality, the ability of proposers to completely control the contents of a block makes censorship relatively fragile. To combat this, a notion of inclusion lists has been proposed in the blockchain community. This paper presents the first formal study of inclusion lists. Our inclusion list design leverages multiple proposers to propose transactions and improve censorship resistance. The design has two key components. The first component is a utility-maximizing input list creation mechanism that allows rational proposers to achieve a correlated equilibrium while prioritizing high-value transactions. The second component, AUCIL (auction-based inclusion list), is a mechanism for aggregating the input lists from the proposers to output an inclusion list.**

1. Introduction

Transaction censorship refers to refusing to process valid transactions for an extended period. Centralized finance services are prone to censorship because important intermediaries, such as banks and exchanges, can heavily influence what transactions are allowed. This power can, and has been, abused to suppress dissent, curb protests, and even harm freedom of speech. For instance, In 2010, the whistleblower website WikiLeaks suffered an extra-legal financial blockade [1]. In 2022, the Canadian government froze the bank accounts of Canadian truck protestors [2]. During the GameStop short squeeze, stock trading app Robinhood barred traders from buying or selling shares [3], allegedly causing millions of dollars in loss [4].

Blockchains such as Ethereum are touted as resilient to censorship, but in reality, the resistance is rather fragile because the creator of a block completely dictates what transactions are allowed in that block. Censorship can be motivated primarily by two reasons. The first reason is regulation. Transaction censorship skyrocketed in 2022 due to an OFAC sanction [5] and affected all users regardless of whether they are subject to US regulations or not. The second reason is financial gains. Suppose T is a user transaction that an adversary wants to censor, and T pays a transaction fee f_T; the adversary just needs to pay the proposer slightly more than f_T to incentivize her to leave T out of their blocks. The rise of Decentralized Finance (DeFi) has created various avenues through which bribed censorship can profit [6], [7], [8].

To tackle this problem, the idea of Inclusion Lists (ILs) is proposed [9], [10], [11]. The high-level idea is to add a new protocol (IL) and modify the blockchain protocol so that a block creator must include transactions output by IL, or the block is considered invalid. The key challenge, of course, is to ensure that transactions sent to IL enjoy stronger censorship resistance than the blockchain itself. A promising technique, e.g., employed by COMIS [12] and FOCIL [13], [14], is to distribute the creation of IL outputs to a set of parties, and the final IL output is a union of each party's output. When properly designed, this can significantly increase the bribery cost because the adversary must now bribe multiple parties simultaneously.

However, we observe that these designs can be problematic when composed with other validity rules of a blockchain protocol, in particular, the block size limit. Most blockchains enforce a block size limit so that blocks can be efficiently processed by the network. These designs, therefore, excuse a block creator from including *anything* in IL if it creates a full block. While full blocks are not common in Ethereum (since its transaction fee mechanism targets half-empty blocks in expectation [15]), other blockchains still suffer from such a scenario being common instead.

In this paper, we propose an IL design that can withstand such adversarial conditions by *requiring* block creators to add transactions in IL, similar to an *unconditional* inclusion list [16]. Our design has two key components: an input list construction phase, where parties compute what inputs yield the highest utility, and an aggregation phase, which combines all input lists into an inclusion list. In the first phase of our design, multiple parties called IL proposers independently create an input list from the mempool. This selection is governed by a utility-maximizing algorithm to achieve a correlated equilibrium, ensuring that IL proposers prioritize high-value transactions without strategic deviation or last-minute changes that could undermine the model's censorship-resistant properties. By coordinating input lists based on a correlated equilibrium, our approach effectively reduces the likelihood of timing games, where proposers might otherwise wait for others' input lists to maximize

their own utility. The correlated equilibrium approach directs each proposer's choices toward an optimal outcome for censorship resistance and transaction inclusion fairness.

Following the construction of individual input lists, an "aggregator" aggregates these lists, combining the IL proposers' submissions into a comprehensive inclusion list. The aggregation process is reinforced by a mechanism we call AUCIL (Auction-based Inclusion List), in which IL proposers compete to construct the largest inclusion list possible. This bidding process ensures that proposers are incentivized to aggregate inclusion lists, ensuring that high-priority transactions appear in the final block.

In summary, this paper:

1) proposes a new definition of censorship resistance for inclusion lists (IL) schemes based on the amount of bribe required to censor each IL output.
2) provides the first formal definition for inclusion lists (ILs), laying the foundation for rigorous treatment of IL proposals;
3) proposes a novel IL protocol combining correlated equilibrium and auction-based inclusion lists (AUCIL);
4) presents a rigorous analysis and shows significant improvement over the state-of-the-art (e.g., FOCIL [13]).

1.1. Design Overview

AUCIL uses a two-phase inclusion list design that maximizes the utility for IL proposers while still guaranteeing strong censorship resistance. The first phase, input list construction, ensures that each IL proposer selects transactions optimally while preventing strategic manipulation. The transaction selection process is governed by a correlated equilibrium strategy, which ensures that the expected utility for each IL proposer is maximized while maintaining censorship resistance. If no predefined strategy were provided, the default behavior of rational IL proposers would be to follow a mixed Nash equilibrium, which theoretically increases censorship resistance but results in lower utility for each proposer in expectation. The correlated equilibrium approach, in contrast, allows proposers to achieve a predictable and utility-maximizing selection, ensuring that their choices do not lead to unnecessary strategic waiting or last-minute adjustments. Among all possible correlated equilibria, the one employed in AUCIL is specifically designed to obtain high censorship resistance while maintaining high proposer utility. In theory, a better design for censorship resistance could be achieved by making the IL Proposer ordering private, which is discussed in Section 7.

The second phase, aggregation, introduces a competitive bidding mechanism to determine the final inclusion list. IL proposers submit their constructed input lists, and all participants can aggregate these lists to form a more comprehensive inclusion list. The auction-based aggregation protocol ensures that the proposer submitting the largest valid inclusion list is rewarded, directly incentivizing proposers to maximize transaction inclusion. To prevent collusion and discourage IL proposers from strategically withholding their input lists, the bidding process incorporates a randomized bias factor generated using Verifiable Random Functions (VRFs). This ensures unpredictability in the auction outcome, making it difficult for the proposers to determine whether to share their input lists.

The design achieves several key properties that enhance censorship resistance and incentive alignment - **(1) Censorship Resistance:** If a transaction is included in the winning inclusion list, the block proposer is required to include it in the block, making transaction exclusion by filling block space infeasible. **(2) Bribery deterrence:** Instead of centrally controlling transaction inclusion, decision-making is distributed across multiple rational IL proposers, forcing an adversary to bribe multiple parties rather than a single entity. The adversary must outbid rational participants with incentives to maximize their rewards. **(3) Verification and enforcement:** To prevent proposer manipulation, verification mechanisms like on-chain validation ensure that the selected inclusion list genuinely reflects (one of) the highest bid. **(4) Crash Fault Tolerance:** The design tolerates crash faults where some parties are unable to participate. **(5) Increased cost of fake crashes:** The block proposer can drop certain bids which would be indistinguishable from the parties never submitting the bids (i.e., crashed). This could lead to an adversary threatening to suppress a non-compliant bid. However, the reward distribution is delayed across multiple rounds, disrupting what could have been an equilibrium of accepting very cheap bribes under threat of suppression.

Combining correlated equilibrium-based selection with auction-driven aggregation, AUCIL ensures that proposers are incentivized to include high-value transactions while maintaining strong censorship resistance. This synergistic mechanism results in an inclusion list protocol that is both rationally optimal for participants and robust against adversarial censorship strategies.

1.2. Paper Outline

Section 2 formalizes censorship resistance and outlines the motivation for using an inclusion list to improve censorship resistance for blockchain protocols. In Section 3, we outline the two-phase protocol that would be followed by AUCIL. In Section 4, we present the first phase of the design, where we propose a greedy allocation of transactions to IL proposers to achieve input lists that follow a correlated equilibrium. The second phase is shown in Section 5, where we use auctions to ensure that maximum input lists get used while aggregating input lists into inclusion lists. Analysis of adversarial censorship attack is shown in Section 6. Section 7 discusses various open problems with the solution, and Section 8 discusses the previous related work on censorship resistance.

2. Problem Statement

The proposed setting consists of two protocols. The first protocol is a **blockchain protocol**, which realizes the abstraction of a state machine replication system. The central

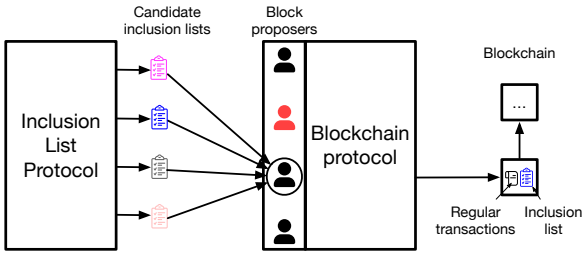


Figure 1: **The setting of an inclusion list with a blockchain protocol.** The inclusion list protocol outputs a set of inclusion lists for each log position, and a block proposed in the blockchain protocol must include one of these lists.

building block of state machine replication is multi-shot consensus, where a set of parties (a subset of which may be faulty) agree on a dynamically growing sequence of log. Such a protocol provides: (i) safety: non-faulty parties agree on each log position, and (ii) liveness: every input that arrives at all parties is eventually recorded in the log. Typically, the log is referred to as a blockchain, and a log entry is referred to as a block. Each block contains a set of transactions that external clients submit to the parties. We refer to the transactions that are not included in the log yet but received by the parties as their mempool. In this paper, we assume a class of blockchain protocols where there is a designated party that is responsible for producing the block at a given position in the log; we refer to these parties as proposers (or leaders).[1] The second protocol is an **inclusion list protocol**, which operates simultaneously with the blockchain protocol. This protocol is run among a set of parties (that may or may not be the same as the ones running the blockchain protocol) that receive transactions in their mempool, and the protocol outputs a set of lists (of transactions). An inclusion list protocol instance is synchronized with the blockchain protocol execution (e.g., running the protocol instance for each position of the log). The key requirement enforced on a blockchain protocol augmented with an inclusion list is that, if an instance for a position j outputs a set S of lists, then the proposer of position j must include some $s \in S$.

Motivation for using an inclusion list. The key motivation for maintaining an inclusion list is censorship resistance. Observe that if all of the parties in the blockchain are honest, then the mempool effectively is an inclusion list. This is because the next proposer will include the set of pending transactions (modulo block size constraints). However, if the proposers are Byzantine or rational, they may not be incentivized to include all of the pending transactions.

The inclusion list is thus used as an enforcement mechanism to ensure that proposers include a specific set of transactions; looking ahead, the goal would be ensure that the inclusion list protocol provides the desired censorship resistance property.

1. We make this assumption for concreteness. At a high level, our result is more generally applicable.

Definition 1 (Inclusion List)**.** *An inclusion list, denoted as IncL, for a log position j is a set of lists of transaction identifiers such that a proposer of log position j must prioritize including transactions of one of the lists in a block within the blockchain protocol.*

In practice, the transaction identifiers could be the transaction hash or a header containing some data that identifies the transaction, like transaction nonce and a sender's address. To simplify, we will assume that the output would contain the entire transaction. We state that transactions in one of the lists is *prioritized* since the list can be longer than the block size. A related definition is that of an unconditional inclusion list, which requires the list to be smaller than the block size.

Definition 2 (Unconditional Inclusion List)**.** *An unconditional inclusion list for a log position j is a set of lists of transaction identifiers such that a proposer of log position j must include the entirety of one of the lists in the block.*

This paper mainly deals with creating an unconditional inclusion list, and any references to the inclusion list would imply an unconditional inclusion list.

Model. All parties in the paper are assumed to be rational, i.e., they try to maximize their utility. The parties' utility is defined as the amount of stake they earn via protocol rewards and external bribes, subtracting protocol-level penalties like slashing. To account for external bribes, we consider an external adversary as a non-participating protocol member with undefined utility from censoring a transaction in the system. The adversary aims to censor a particular transaction (which we will refer to as a target transaction) while minimizing the cost incurred. This cost is incurred when the adversary bribes a rational party into an action that otherwise reduces the said party's utility. In practice, the adversary may control some consensus parties, but to simplify the context, we can assume the controlled nodes to be rational and bribed. Throughout the paper, we assume that all parties are in sync, i.e., all parties have the same mempool, and the network is also synchronous.

Censorship resistance. As described earlier, the key reason for introducing an inclusion list in a blockchain is censorship resistance.

Definition 3 ((B, θ, T)-Censorship Resistance)**.** *Given an adversary with an arbitrary bribing budget b, a protocol running with a mempool M is said to be (B, θ, T)-censorship resistant for a set of target transactions $T(M)$ if there exists a set of at least θ parties such that for all $i : t_i \in T(M), b_i \in B$ if $b < b_i$ all parties in this set include the transaction t_i in their output.*

In other words, for any transaction $t_i \in T(M)$, the same set of at least θ parties must include each transaction t_i in their respective outputs unless the bribe budget exceeds the tolerated bribe limit b_i. In the definition, we have three parameters: the tolerated bribe limit array B, the number of parties θ, and a function T of mempool for transactions. The

tolerated bribery budget $b_i \in B$ represents the maximum amount of bribe an adversary can spend to exclude some target transaction $t_i \in T(M)$. This includes the cost of bribing the rational parties and alternative ways, like introducing additional transactions to modify the mempool. θ represents the minimum number of parties that output the same set of transactions. This parameter is required to tolerate $\theta - 1$ crash faults that occur among the parties in the protocol.

Example 1 (Censorship resistance on an input)**.** *Consider the following example. There exist six transactions* $M = \{m_1, m_2, m_3, m_4, m_5, m_6\}$ *paying a fee of* $\{10, 9, 8, 3, 2, 1\}$ *respectively. Consider a protocol with no crash faults (*$\theta = 1$*) that can output a set of at most 4 transactions. Then a meaningful* $T(M)$ *could be* $\{m_1, m_2, m_3, m_4\}$ *which tolerates a bribery budget of* $B = \{10, 9, 8, 3\}$ *respectively. An alternate* $T(M)$ *could be* $\{m_1, m_2, m_3\}$*, which provides a different tolerance of bribery as* $B = \{20, 9, 8\}$*. Both such designs would follow their respective definition of* (B, θ, T)*-Censorship Resistance.*

This paper presents a protocol design in which at least θ different parties generate an inclusion list $T(M)$ while tolerating a bribery limit of B, thus providing (B, θ, T)-censorship resistance. The **aim** is two-fold: to find $T(M)$ such that high fee-paying transactions are preferred while maintaining high throughput and the tolerated bribe for each of these transactions is proportional to n and f_i, where f_i represents the fee paid by each transaction tx_i and n is the total number of parties in the system.

Before discussing the protocol in detail, we introduce some recurring terms in the paper. Each party chosen to add transactions to a candidate inclusion list (in the set of output lists) is referred to as an *IL Proposer*. We refer to the input of each IL Proposer as an *input list*. Looking ahead, these lists will be aggregated with other input lists from other IL proposers to create a candidate inclusion list. Previous research [12], [13], the term Local Inclusion List was used to refer to the inputs of IL proposers. We deviate from this terminology since input lists for our protocol do not resemble the output inclusion lists in many cases. Contrary to previous work, since this paper deals with unconditional inclusion lists, we restrict the size of each input list as k, such that $n \cdot k \leq$ block size.

3. Protocol Outline

Consider a set of m transactions, $M = \{m_1, \ldots, m_m\}$, and n IL proposers, $N = \{P_1, \ldots, P_n\}$. The IL Proposers are chosen through some randomized committee selection amongst a set of staked validators. Each IL proposer P_i proposes an input list $\text{InpL}_i = \{m_1^i, \ldots, m_k^i\}$, where each $m_j^i \in M$. The inclusion list would be computed as a subset of the union of all input lists. This role of computing a union would be assigned to a set of aggregators. An aggregator could be a single chosen party, a set of parties, or all parties. This paper will present a design where all parties behave as aggregators, out of which the blockchain proposer chooses one. Thus, the design in the paper will follow the following outline:

- All IL proposers create an input list, InpL, each from a set of transactions in their mempool (Section 4).
- IL proposers send their InpL to the aggregators. Each aggregator collects all the InpLs and computes a union to create an inclusion list, IncL (Section 5).
- The block proposer of the next block in the blockchain protocol must ensure the inclusion of all transactions in one of the inclusion lists.

Under a rational threat model, it is important to have a robust incentive scheme. Each transaction tx_i has a censorship-resistance fee f_i. We define a contribution score CS_i^j for each transaction tx_i and IL proposer P_j representing how much the IL proposer contributed towards the addition of transaction in the inclusion list created. Thus, each IL proposer P_j would receive a share $\text{CS}_i^j \cdot f_i$ for each transaction tx_i included in the inclusion list. Some required properties from contribution score allocations would be:

- No contribution score can be negative, i.e., $\text{CS}_i^j \geq 0 \; \forall i, j$.
- The sum of the contribution score for each transaction should be 1, i.e., $\sum_{j,\ P_j \in N} \text{CS}_i^j = 1$
- Given two IL proposers taking the same action and contributing equally (on-chain) to the inclusion of a transaction, then their contribution score would be the same.

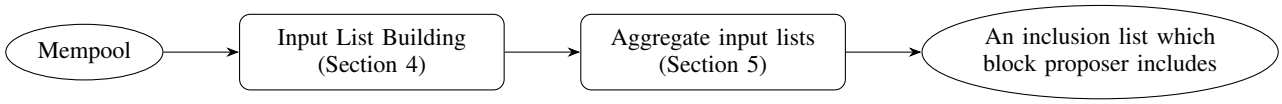
Based on this protocol outline, we present our solution in two phases - the input deciding phase (Section 4) and then the aggregation phase (Section 5).

4. Input List Building Protocol

This section presents a protocol for how an IL proposer should build an input list.

4.1. A Naïve Approach

Consider that the only action an IL proposer can take to increase its contribution score for a transaction is to include it in the input list. Thus, given the required properties of the contribution score, the only satisfying contribution score mechanism would be if the fee is divided equally amongst all parties that include the transaction in their input list, and the input list is included in the set of final inclusion lists. Let us consider IL proposers naïvely choosing the transactions that pay the highest fee independent of what other IL proposers might choose. This naïve scheme of greedily picking transactions without considering other parties' actions is not a Nash equilibrium. Given all other IL proposers' input lists that consist of greedily selected transactions, the rational choice for an IL proposer may not be to construct its own input greedily. The example in Table 1 confirms the stipulation.



Strategy	Objects Picked	Utility
Pick Top Paying	m_1, m_2	7
Alternate	m_3, m_4	15

TABLE 1: Picking top-paying objects is not a Nash equilibrium. Consider three parties ($n = 3$), selecting two transactions ($k = 2$) each from $\{m_1, m_2, m_3, m_4, m_5, m_6\}$ with utilities $\{11, 10, 9, 6, 4, 3\}$ respectively. Other parties are assumed to follow the strategy of picking the top-paying transaction.

A natural equilibrium would be to consider mixed strategies. In this, each party would choose a transaction with some pre-defined probability. In this case, it would be $f_i/\sum(f_i)$ for a transaction tx_i paying a fee f_i (More details in Appendix A) such that selecting transactions from outside the probability distribution would result in lower expected revenue. However, we cannot rely on a mixed Nash equilibrium as is (without the use of commit and reveal primitives). This is because parties may engage in a timing game. Parties can wait for others to broadcast their input lists before creating their own. This creates a complex game where actions are influenced by the time taken to broadcast the input list.

4.2. Input List Building Protocol

Can we accomplish an equilibrium where the parties can predict what others will propose before the broadcast? To answer this, we use the notion of *correlated equilibrium* where parties are suggested (by a third party) to take a particular action. This action corresponds to the maximum utility that can be received by such a party, given that all the other parties follow the suggested action.

The idea behind using a correlated equilibrium is simple. If you know what the other parties will do, there is no point in waiting for them to produce and declare their input. Let us define this game more formally. Consider a setting with m objects (transactions) and n parties (IL proposers). Each party can select a maximum of k distinct objects. Each object m_i is associated with a utility value u_i, which is uniform across all parties. When multiple parties select an object, the utility u_i is shared (split) equally among the selecting parties. Further, after allocation, with probability $1-\gamma$, the allocation for each party is dropped (and given an empty set instead). The reason for this probabilistic dropping will beclear in the next section. The objective for each party is to maximize its expected utility. We assume that all IL proposers value transactions solely based on the fees they offer, i.e., $u_i = f_i$.

Define n_i as the number of parties selecting object m_i and N_i as the set of parties that have chosen m_i. The expected number of parties that select the object m_i (and don't drop later) is γn_i. Let L_j denote the set of objects allocated to party P_j, and let $L = \{L_1, \ldots, L_n\}$ represent an allocation satisfying the problem's constraints. We define $\mathbb{U}(L_j)$ as the utility derived by party j from selecting objects in L_j given all other parties accept the allocation in L.

$$\mathbb{U}(L_j) = \sum_{m_i \in L_j} \frac{u_i}{\gamma(n_i - 1) + 1} \tag{1}$$

Notice the denominator. The reason for using this is that while for an external party viewing the problem, the expected number of parties that select the object is γn_i, for the party choosing it, the expected number of other parties, apart from itself, would be $\gamma(n_i-1)$. Now, for this party, the probability of whether it is dropped doesn't affect the utility it receives, and thus, the total number of parties, if this party receives any utility, would be $1+\gamma(n_i-1)$. For ease of notation, let $\widetilde{n_i}$ represent the denominator ($\widetilde{n_i} = 1+\gamma(n_i-1)$).

We aim to reach a correlated equilibrium for the above game. An external party—in this case, the protocol—recommends which objects to choose for each party in a correlated equilibrium. This subsection aims to replace this external party with an algorithm publicly known to all parties. All parties observe their own set of objects and the sets of all other parties' sets of objects as recommended by the algorithm. Given this information, the party should have no incentive to deviate from the recommended objects.

Let us represent the constraints mathematically:
The number of times each object can be selected is limited by the number of total parties.

$$\forall m_j \in M : n_j \leq n \tag{2}$$

Next, we constrain that each party can be allocated at most k objects.

$$\forall L_i \in L : |L_i| \leq k \tag{3}$$

The last constraint we want to represent is for achieving correlated equilibrium. Suppose L is suggested to all parties. Correlated equilibrium states that choosing any other set of objects would lead to a utility less than or equal to the utility from the set that the algorithm recommends. Formally, let $M^{[\leq k]}$ represent a subset of $M = \{m_1, \ldots, m_m\}$ with cardinality less than or equal to k, we require

$$\forall L'_i = M^{[\leq k]}, L_i \in L : \mathbb{U}(L_i) \geq \mathbb{U}(L'_i) \tag{4}$$

Let $T(M)$ represent all the objects the protocol assigns to at least one party $T(M) = \bigcup L_i$. Consider the following constraint:

$$\forall m_a \in T(M), m_b \in M : \\ \exists i : (L_i \in L,\ m_a \in L_i, m_b \notin L_i) \implies \frac{u_a}{\widetilde{n_a}} \geq \frac{u_b}{\widetilde{n_b} + \gamma} \tag{5}$$

In words, this constraint says that if there exists an object allocated to some party and another object not allocated to the same party, then for the party, swapping the objects cannot lead to a better utility. This is because swapping would increase the expected number of times the object appears by γ from the perspective of the party.

Lemma 1. *If $|M| < k$, then $L_i = M$.*

Consider to the contrary, $|L_i| < |M|$. In such a case, there exists an object m_j not in the allocation L_i. Since L_i is a subset of M, $|L_i| < k$, and thus the object L_i can be added to the allocation.

Lemma 2. *If $|M| \geq k$, the constraint* (5) *implies constraint* (4)*.*

Proof. Since $|M| \geq k$, $|L_i| = k$. This is because if $|L_i| < k$, then there exists an object in M, not in L_i, which can be added to increase the utility gained by the selection. Now let's assume, to the contrary, that constraint (5) holds but constraint (4) does not. The negation of constraint (4) states

$$\exists L_i \in L, \exists L_i' = M^{[\leq k]} : \mathbb{U}(L_i) < \mathbb{U}(L_i')$$

Let $L_i' \neq L_i$ represent the allocation with the highest utility (strictly greater than allocation L_i). If multiple such allocations exist with the highest utility, consider L_i' as the set that differs in the least number of objects from L_i. There exists an object m_a such that $m_a \in L_i$ but $m_a \notin L_i'$. (If no such object exists, then $L_i' \supseteq L_i$. However since $|L_i'| \leq k = |L_i|$, this is possible only if $L_i = L_i'$)

Consider $|L_i'| < k$. Adding m_a would only add to the party's utility; however, L_i' was considered as the maximal such allocation, and thus such an L_i' can only exist if $|L_i'| = k$. Since both L_i' and L_i are of size k and there exists $m_a \in L_i$ but $m_a \notin L_i'$, there must exist $m_b \in L_i'$ but $m_b \notin L_i$.

Given all other allocations $L_j \neq L_i$ remain the same, the number of times the object m_b is chosen increases by 1 in L_i' compared to L_i. From(1),

$$\mathbb{U}(L_i) = \mathbb{U}(L_i \setminus m_a) + \frac{u_a}{\widetilde{n_a}}$$
$$\mathbb{U}(L_i') = \mathbb{U}(L_i' \setminus m_b) + \frac{u_b}{\widetilde{n_b} + \gamma}$$

From constraint(5), in this case (where $\exists i : (L_i \in L, m_a \in L_i, m_b \notin L_i)$),

$$\frac{u_a}{\widetilde{n_a}} \geq \frac{u_b}{\widetilde{n_b} + \gamma}$$

Consider the set $L_i'' = (L_i' \setminus \{m_b\}) \cup \{m_a\}$. $|L_i''| = |L_i'|$, and thus satisfies $M^{[\leq k]}$ property. The utility of this set is

$$\begin{aligned}\mathbb{U}(L_i'') &= \mathbb{U}(L_i' \setminus m_b) + \frac{u_a}{\widetilde{n_a}} \\ &= \mathbb{U}(L_i') + \frac{u_a}{\widetilde{n_a}} - \frac{u_b}{\widetilde{n_b} + \gamma} \\ &\geq \mathbb{U}(L_i')\end{aligned}$$

This is a contradiction since L_i' was considered as the set with the highest utility that differed in the least elements compared to L_i; however, we show the existence of another set with one less element differing from L_i, which has a utility greater than or equal to L_i'. □

The above set of constraints forms the basis of our problem. Any solution that satisfies the above constraints can be used as a third party that informs the IL proposers of the transactions to pick.

4.3. A Greedy Algorithm

If we use the above constraints as invariants in a greedy, step-by-step selection and then allocate selected objects in a round-robin allocation, the final allocation would satisfy the required constraints by default.

Algorithm 1 presents a greedy algorithm that chooses and assigns the objects to parties while satisfying the invariants defined in Eq (2), (3) and (5). To do so, we choose objects one at a time, picking the object with the highest local utility $U \oslash N$, where $\oslash$ denotes element-wise division of the utility by the number of times the object would have been picked if selected now (in expectation). By picking the object, we increase the number of times it is selected by the probability that this selection of the object is broadcast. This updates N such that $U \oslash N$ represents the utility when the object is next selected. Next, if the object has been selected n times (making $N[i] = n\gamma + 1$), the utility is set as negative to ensure that the object is not picked again (since all parties would be assigned this object).

For each object, such that $N[i] \neq 1$, i.e., the number of times it has been selected is at least one, the invariant in Eq (5) is maintained. The constraint Eq (2) is satisfied by setting the object's utility to -1 after the object is selected n times and ensuring it is never picked again.

Having chosen a list of objects and decided the number of times they will be chosen, the second step is to assign/allocate them to parties. Any allocation that assigns objects uniquely to all parties and ensures that all parties receive the same number of objects (at most k objects are assigned per Eq (3)) would satisfy the correlated equilibrium since all the constraints are satisfied. However, to allocate fairly, we use a round-robin allocation, assigning the highest-valued object one at a time to each party. The fairness guarantees are shown in Section D.

In the following theorem, we show that the above algorithm gives a correlated equilibrium.

Theorem 1. *[Correlated Equilibrium] Given the assignment of input lists according to Algorithm 1, if every other party follows the assignment, then no party obtains a better utility by swapping any of the assigned objects in its input list with any other object not on its list.*

To understand the proof of the above theorem, consider the following. Let some object o_i be picked for the last time (i.e. the utility after this pick does not decrease for o_i). For this object, the utility to whichever party it is given to is greater than all other candidate objects. Further, the utility of other objects could decrease, but the utility for this object does not decrease (since we are considering the last instance

Algorithm 1 A 2-Step algorithm for transaction inclusion

Require: $n \geq 0$, $m \geq 0$, $k \geq 0$, $\gamma \geq 0$
▷ number of parties, transactions, input list size, probability of broadcast

Ensure: L_i arrays for all $i \in P$
▷ final inclusion arrays for each party

Step 1: Choose Objects

1: $U \leftarrow [u_1, \ldots, u_m]$ ▷ Utility values for each transaction
2: $N \leftarrow [1, \ldots, 1]$ ▷ Count array (corresponding to $\widetilde{n_i}$), initialized to 1 for each transaction
3: $S \leftarrow \{\}$
4: **for** $\_\_ \in 1$ to $n * k$ **do**
5: $U_{curr} \leftarrow U \oslash N$ ▷ Compute current utility by element-wise division of utility by the expected number of times the object has been selected
6: $s \leftarrow \text{argmax}(U_{curr})$ ▷ Find the index of the maximum value in U_{curr}
7: $S \leftarrow S \cup \{s\}$
8: **if** $U[s] = -1$ **then** break ▷ if the maximum utility for objects is -1, then all objects have been selected n times (Line 10 sets -1 for objects selected n times)
9: $N[s] \leftarrow N[s] + \gamma$ ▷ Increment the expected number for the next time this object is selected
10: **if** $N[s] + \gamma \geq n\gamma + 1$ **then** $U[s] \leftarrow -1$ ▷ Set utility of object to -1 if it has been allocated n times

Step 2: Allocate Objects

11: $U \leftarrow [u_1, \ldots, u_m]$ ▷ Reset utilities
12: $P \leftarrow [1, \ldots, n]$ ▷ Array of party identifiers
13: $\forall i \in P : L_i \leftarrow \{\}$ ▷ Inclusion sets for each party, initialized to empty
14: $U_f \leftarrow U \oslash_s (N \ominus [\gamma]^m)$ ▷ $\oslash_s$ is element-wise safe division, which returns 0 if dividing by 0. This computes the utility for each selected object, adjusting for the extra γ added for the preparation of the next selection
15: $A \leftarrow sort(S, \text{key} = (-U_f[s] \text{ for } s \in S, s))$ ▷ Get indices of objects in descending order of utilities ($[U_f[A[0]], U_f[A[1]], \ldots]$ is in descending order)
16: **for** $j \in 1$ to $|A|$ **do**
17: $L_{(j \bmod n)+1} \leftarrow L_{(j \bmod n)+1} \cup \{A[j]\}$ ▷ Assign objects in round-robin fashion
18: **return** $\forall i \in P : L_i$ ▷ Return the inclusion sets for all parties

Description: This algorithm iteratively selects objects with the highest available utility. After all objects have been selected, they are assigned to parties in a round-robin format, with the highest utility object being assigned first. A follow-along example is shown in Appendix C.

of selection for the object). Thus, if after the assignment is complete, any party tries to switch away from object o_i to any other object o_j, the utility would be lower.

Proof. We prove the theorem by contradiction. Let $U(o)$ represent the from object o in the given assignment; $U^*(o)$ represent the utility from object o if it is chosen one extra time; and $\bar{U}(o, l)$ represents the utility for selecting object o in round l. Let some party have a positive incentive (> 0) to switch from object o_i to o_j. Let's say the object o_n is selected as the final selection of the algorithm (Last round of Step 1).

First, note the following

$$U^*(o) = \bar{U}(o, n \cdot k) \ \forall o \in M \setminus \{o_n\}$$
$$\bar{U}(o, l) \geq \bar{U}(o, n \cdot k) \ \forall o \in M$$

Also, choosing o_n in the last round implies that its utility would be lower if it chose any object except o_n. In particular,

$$\bar{U}(o_n, n \cdot k) \geq \bar{U}(o; \forall o, n \cdot k) \geq U^*(o) \ \forall o \in M \setminus \{o_n\}$$

Let l be the round in which object o_i was last chosen.

$$\bar{U}(o_i, l) \geq \bar{U}(o, l) \ \forall o \in M$$
$$\bar{U}(o_i, l) = U(o_i)$$

Thus,

$$U(o_i) \geq \bar{U}(o, l) \geq U^*(o) \ \forall o \in M$$

Now, the utility to swap $o_i \rightarrow o_j$ is

$$U(swap(o_i, o_j)) = U^*(o_j) - U(o_i) \leq 0$$

Thus, there is a contradiction since the utility from the swap needed to be > 0. □

5. Aggregation Protocol

In Section 4, we presented a protocol for IL proposers to build their input list and achieve a correlated equilibrium. The next step in the design is to aggregate these input lists into inclusion lists, which would be used to constrain the builder. In this section, we will present a protocol to aggregate the input lists generated by parties such that the overall inclusion list design outputs a set of candidate lists.

Key design challenges. There are two key challenges we need to address. First, since the lists are being aggregated, the adversary can censor a target transaction by somehow ensuring that the input lists containing the target transaction are not aggregated. This can be achieved, for example, by bribing a party that aggregates.

Second, some of the parties in the protocol may crash, and the protocol needs to tolerate certain absence of input lists. Suppose the aggregator is required to include all transactions. In that case, even if one IL proposer goes offline or chooses not to broadcast, an honest aggregator cannot create an inclusion list by including all other input lists. Thus, our crash-tolerant protocol would only require the aggregator to include all but a threshold θ of the input lists. However, in

doing so, we allow the aggregator to censor complete input lists willfully without any penalties. Thus, in this section, we will describe a protocol that the aggregator and IL proposers must follow in addition to following the correlated equilibrium suggestion in Section 4.2. To incentivize the parties to follow the protocol, we describe a reward distribution mechanism that maximizes the cost for the adversary to exclude a transaction from the inclusion list.

5.1. Outlining our Solution: AUCIL

The design we will discuss is inspired by a winner-take-all game like an auction, where the bid submitted is the length of the inclusion list. We name this design AUCIL.[2] In AUCIL, all parties work as aggregators. All IL proposers can aggregate the input lists created and broadcasted by IL proposers. After aggregation, each party declares their bids as the size (in terms of the number of input lists included) of the inclusion list they created. A natural way of collecting these bids is through the block proposer of the next block in the blockchain. This block proposer would accept all bids and add the inclusion list with the highest bid.

Algorithm 2 AUCIL outline

Participants: All IL proposers $P_1, P_2, \ldots, P_n$

Step 1: IL proposers broadcast input lists

1: **for all** P_i **do**

2: $P_i \rightarrow_B$ all parties : InpL_i

Step 2: Parties aggregate input lists into an inclusion list and broadcast it

3: **for all** P_j **do**

4: $\text{IncL}_j = \bigcup_{i=1}^{n} \text{InpL}_i$

5: $P_j \rightarrow_B$ all parties : $(\text{IncL}_j, \ell_j = \text{size}(\text{IncL}_j))$

Step 3: Proposer selects the highest bid inclusion list

6: Proposer receives:

$$\{(\text{IncL}_1, \ell_1), (\text{IncL}_2, \ell_2), \ldots, (\text{IncL}_n, \ell_n)\}$$

7: Proposer selects the highest bid where ℓ_i denotes the bid for P_i.

Incentive Structure: IL Proposer of the selected bid receives u_{agg}

As described in Algorithm 2, the outline has three major steps – (i) InpL broadcast, (ii) bid creation and broadcast, and (iii) collection of bids. While the second step is a competition between bidders and is thus incentivized by a reward for winning (u_{agg}), the other two steps are not incentivized. For the first step, what is the incentive for each IL proposer to share their input list? If all parties share the input lists and the proposer picks the inclusion list with the highest bid, it is strictly dominant for a party not to share its input list since it could aggregate others' input lists with its own and create the highest-size inclusion list, and thus win the auction. Thus, sharing the input list is not an equilibrium for such a party.

Moreover, even if all parties declare their input lists, the auctions also suffer from censorship problems, as also highlighted in [8]. The next block proposer could ignore the competitive bids in favor of an adversarial bid, which bribes the proposer to exclude other bids. However, distinct from on-chain auctions, this off-chain auction has the essential property of having a fixed number of bidders, all of which have a positive utility to bid. While some parties could crash - or choose to feign a crash as an adversarial action - at least a threshold $n - \theta$ would (or have the option to) broadcast their bid, and the block proposer could be required to prove that the bid included is greater than at least $n-\theta$ other bids. If the proposer cannot create such a proof, then the block generated would be invalid, and the block proposer would be slashed. This proof would be verified on-chain (by the consensus as a part of validity condition or by using fraud proofs [17]).

Given this threshold, θ, of parties whose bids are not required to create an inclusion list, the adversary could censor some bids that include the target transaction after the bids have been sent. To incentivize the bids to be as high as possible, despite the censorship, the reward distribution for aggregation (u_{agg}) would take place some blocks after the block for which the inclusion list is being designed. The bids would continue to be collected in the next round(s). The rewards would be distributed as ($u_{agg}/2$) for the highest bid (observed across multiple blocks) and the rest ($u_{agg}/2$) for the winning bid (observed during the block for the creation of the inclusion list). This distribution of rewards is arbitrarily chosen, and other reward distributions could exist.

To solve the first problem, let's consider the other extreme situation. If no party shares its input list, it is dominant for most IL proposers to share it. This is because it would not be able to create a winning bid; if the tie is broken randomly, there is a $1/n$ probability of winning. Releasing its input list thus allows the winning bid to include this input list and thus give a transaction fee reward as described in Section 4. This proves that the Nash equilibrium for the above game is mixed and probabilistically lies somewhere between sharing the input list and not sharing the input list. The exact equilibrium would depend on the probability of the IL proposer winning the game by not sharing the input list.

Herein lies the solution. If the probability of winning the auction is low for most parties in the system, then their dominant action is to broadcast their input list. Thus, we bias the auction results so that most parties will lose the auction with a high probability. We introduce a local noise value, which we call bias b, drawn uniformly at random from the range $[0, b_{max}]$ and add it to the bid; this can be obtained using VRFs [18]. The bid now becomes the length of the inclusion list plus the random noise. If a party draws a low random noise, then it is highly likely that the party would lose the auction, and thus, the party would have a high incentive to release its input list. We use this to complete the AUCIL design in algorithm 3.

However, this does not remove the case where parties do not broadcast their input list; for high bias values, AUCIL encourages them not to broadcast their input lists. In order to separate the notion of broadcasting the input list and

2. AUCIL stands for Auction-based Inclusion List.

others adding it to their bid, we introduce a metadata value associated with the input list, which we label as flag **F**. If **F** is set to 1, then the input list is available to all parties and increases the bid size by 1 when included in the bid. If **F** is set to 0, then even if the input list is included in a bid, it would not increase the bid size. In return, only those input lists that have **F** set to 1 would receive their share of transaction inclusion rewards as described in the previous section. In a rational world, setting the flag to 1 is equivalent to broadcasting, and 0 is the same as withholding its input list. In further sections, an input list is considered available if it has been broadcast and **F** is set to 1. A more formal analysis of the probability of a party's releasing its transaction follows.

5.2. Analysis

In this section, we will analyze the utility of IL proposers in the absence of adversarial censorship. In such a case, the proposer will select the highest bid amongst all the bids. The first thing to observe is that not all parties may have an incentive to broadcast, so let's assume that η (At equilibrium, $\gamma = \eta/n$) InpLs are publicly available. Consider that the IL proposer includes all the input lists it receives and its own. The IL proposer has two options: broadcast its InpL or withhold it. If the IL proposer broadcasts its block, then the following lemma holds.

Lemma 3. *Given an IL proposer P with a utility u_{il} for inclusion of its input list and u_{agg} for winning the auction for aggregation. Given η input lists are available (except its own), and the total number of IL proposers is n. Given P generates a bias $b \leq 1$. If P chooses to make its InpL available, its expected utility is $u_{il} + \mathsf{negl}(n)$.*

Proof. The bid generated by P is calculated as $\eta + 1 + b$. All other IL proposers also receive the input list of P. There exist two classes of other IL proposers: 1) those that made their input list available (there exist η such IL proposers) and those that did not $(n - \eta - 1)$. Let b_i represent the bias generated through VRF for IL proposer P_i.

The bid for each IL proposer who did not make its input list available is $\eta + 2 + b_i$ ($\eta + 1$ from publicly available lists, and 1 private). Similarly, the bid for each IL proposer who chose to make its input list available is $\eta + 1 + b_i$ (Its list is included in the publicly available lists).

The probability that P wins the auction is the same as the bid generated by P being greater than all other bids.

$$\begin{aligned}
\mathbb{P}(P \text{ wins}) &= \prod_{i=0}^{\eta} \mathbb{P}(\eta + 1 + b \geq \eta + 1 + b_i) \\
&\quad \cdot \prod_{i=0}^{n-\eta-1} \mathbb{P}(\eta + 1 + b \geq \eta + 2 + b_i) \\
&= \prod_{i=0}^{\eta} \mathbb{P}(b \geq b_i) \cdot \prod_{i=0}^{n-\eta-1} \mathbb{P}(b \geq 1 + b_i) \\
&= \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \prod_{i=0}^{n-\eta-1} \{\mathbb{P}(b \leq 1)\mathbb{P}(b \geq 1 + b_i | b \leq 1) \\
&\quad + \mathbb{P}(b > 1)\mathbb{P}(b \geq 1 + b_i | b > 1)\} \\
&= \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \prod_{i=0}^{n-\eta-1} (\mathbb{P}(b > 1)\mathbb{P}(b - 1 \geq b_i | b > 1)) \quad (6)
\end{aligned}$$

If $b \leq 1$, the probability of winning the auction is 0, unless all parties $(\eta = n-1)$ make their input lists available.

The utility in this case is given by u_{il}. If all parties make their list available, then the utility would increase by $(\frac{b}{b_{max}})^n \cdot u_{agg}$, which is negligible in n. □

Lemma 4. *Given an IL proposer P with a utility u_{il} for inclusion of its input list and u_{agg} for winning the auction for aggregation. Given η input lists are available (except its own), and the total number of IL proposers is n. Given P generates a bias $b > 1$. If the IL proposer chooses to make its InpL available, then its expected utility is $u_{il} + \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \left(\frac{b-1}{b_{max}}\right)^{n-\eta-1} u_{agg}$*

Proof. From (6), the probability of winning the auction is

$$\mathbb{P}(P \text{ wins}) = \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \left(\frac{b-1}{b_{max}}\right)^{n-\eta-1}$$

If P wins the auction, then it will receive both input list inclusion and aggregation rewards, while if it loses the auction, then the reward earned is only the input list inclusion reward.

$$\begin{aligned}
u_P &= \mathbb{P}(P \text{ wins})(u_{agg} + u_{il}) + (1 - \mathbb{P}(P \text{ wins}))\, u_{il} \\
&= u_{il} + \left(\frac{b}{b_{max}}\right)^{\eta} \cdot \left(\frac{b-1}{b_{max}}\right)^{n-\eta-1} u_{agg} \qquad (7)
\end{aligned}$$

□

Lemma 5. *Given an IL proposer P with a utility u_{il} for inclusion of its input list and u_{agg} for winning the auction for aggregation. Given η input lists are available (except its own), and the total number of IL proposers is n. Given P generates a bias $b < b_{max} - 1$. If the IL proposer chooses not to make its InpL available, then its expected utility is $\left(\frac{b+1}{b_{max}}\right)^{\eta} \cdot \left(\frac{b}{b_{max}}\right)^{n-\eta-1} (u_{agg} + u_{il})$*

Proof. The bid generated by P is $\eta + 1 + b$. All other IL proposers can not extend their bid with the input list of

Algorithm 3 AUCIL for log position pos

Participants: All IL proposers $P_1, P_2, \ldots, P_n$

Step 0: IL proposers generate their auction bias

for all P_i **do**

$P_i : (b_i, \pi_i) \leftarrow \text{VRF}_{sk_i}(\text{pos})$ scaled to a range of $[0, b_{max}]$ ▷ Generate the random bias (uniform dist.) between 0 and b_{max} to add to the bid

Step 1: IL proposers broadcast input lists

for all P_i **do**

$P_i : \mathbf{F}_i \leftarrow \text{checkAvailable}(b_i)$ ▷ Check whether P_i should make its input list available

$P_i \rightarrow_B$ all parties : $\text{InpL}_i, \mathbf{F}_i$ ▷ Parties broadcast their InpL while choosing to make it available or not

Step 2: Parties aggregate input lists into an inclusion list and broadcast it

for all P_j **do**

$\text{IncL}_j, y_j = \bigcup_{i=1}^{n} \text{InpL}_i, \sum_{i=1}^{n} \mathbf{F}_i$ ▷ If some IL Proposer's value is missing take $\{\}, 0$ as InpL and $\mathbf{F}$

$P_j \rightarrow_B$ all parties : $(\text{IncL}_j, \ell_j = (y_j + b_j))$ ▷ Parties declare their bid with added bias

Step 3: Block proposer selects the highest bid inclusion list

Proposer receives: $\{(\text{IncL}_1, \ell_1), (\text{IncL}_2, \ell_2), \ldots, (\text{IncL}_n, \ell_n)\}$

Proposer selects the highest bid and adds it to the block (IncL, ℓ).

Proposer verifies π_i, includes proof for the bid being greater than $n - \theta$ bids.

- Block is considered verified if the proof is valid.

Description: Algorithm for aggregating input lists into an inclusion list. The basis of AUCIL is an auction design, where all parties try to compete with the largest size input list. Even after the next block is created, the bids are still collected in case a bid higher than the winning bid is found.

Incentive Structure: IL Proposer of selected bid receives $0.5u_{agg}$. IL Proposer of highest bid across multiple slots receives $0.5u_{agg}$.

P. There exist two classes of other IL proposers - those that made their input lists available (there exist η such IL proposers) and those that did not $(n-\eta-1)$. Let b_i represent the bias generated through VRF for IL proposer P_i.

The bid for each IL proposer who made their input list available is $\eta + b_i$. Similarly, the bid for each IL proposer who did not is $\eta + 1 + b_i$.

The probability that P wins the auction is the same as the bid generated by P being greater than all other bids.

$$\begin{aligned}\mathbb{P}(P \text{ wins}) &= \prod_{i=0}^{\eta} \mathbb{P}(\eta + 1 + b \geq \eta + b_i) \\ &\quad \cdot \prod_{i=0}^{n-\eta-1} \mathbb{P}(\eta + 1 + b \geq \eta + 1 + b_i) \\ &= \prod_{i=0}^{\eta} \mathbb{P}(b + 1 \geq b_i) \cdot \prod_{i=0}^{n-\eta-1} \mathbb{P}(b \geq b_i)\end{aligned}$$

$$\begin{aligned}\mathbb{P}(b + 1 \geq b_i) &= \mathbb{P}(b \leq b_{max} - 1)\mathbb{P}(b + 1 \geq b_i | b \leq b_{max} - 1) \\ &\quad + \mathbb{P}(b > b_{max} - 1)\mathbb{P}(b + 1 \geq b_i | b > b_{max} - 1)\end{aligned}$$

If $b > b_{max} - 1$, then $b + 1$ is always $> b_i$. Thus,

$$\begin{aligned}\mathbb{P}(b + 1 \geq b_i) &= \mathbb{P}(b \leq b_{max} - 1)\mathbb{P}(b + 1 \geq b_i | b \leq b_{max} - 1) \\ &\quad + \mathbb{P}(b > b_{max} - 1) \qquad (8)\end{aligned}$$

Since $b < b_{max} - 1$,

$$\begin{aligned}\mathbb{P}(b + 1 \geq b_i) &= \mathbb{P}(b + 1 \geq b_i | b \leq b_{max} - 1) \\ &= \left(\frac{b+1}{b_{max}}\right)\end{aligned}$$

Similarly,

$$\mathbb{P}(b \geq b_i) = \frac{b}{b_{max}}$$

Thus, given $b \leq b_{max} - 1$, the probability of winning the auction is

$$\mathbb{P}(P \text{ wins}) = \left(\frac{b}{b_{max}}\right)^{n-\eta-1} \cdot \left(\frac{b+1}{b_{max}}\right)^{\eta}$$

If P wins the auction, then it will receive both input list inclusion and aggregation rewards, while if it loses the auction, then no reward is earned since the input list was not available to others.

$$\begin{aligned}u_P &= \mathbb{P}(P \text{ wins})(u_{agg} + u_{il}) \\ &= \left(\frac{b+1}{b_{max}}\right)^{\eta} \cdot \left(\frac{b}{b_{max}}\right)^{n-\eta-1} (u_{agg} + u_{il})\end{aligned}$$

□

Lemma 6. *Given an IL proposer P with a utility u_{il} for inclusion of its input list and u_{agg} for winning the auction for aggregation. Given η input lists are available (except its own), and the total number of IL proposers is n. Given P generates a bias $b \geq b_{max} - 1$. If the IL proposer chooses*

not to make its InpL available, then its expected utility is $\left(\frac{b}{b_{max}}\right)^{n-\eta-1}(u_{agg}+u_{il})$

Proof. Equation (8) still holds for the analysis of this Lemma. Given $b > b_{max}-1$, we get

$$\mathbb{P}(b+1 \geq b_i) = 1$$

Thus,

$$\begin{aligned}\mathbb{P}(P \text{ wins}) &= \left(\frac{b}{b_{max}}\right)^{n-\eta-1}\\ u_P &= \mathbb{P}(P \text{ wins})(u_{agg}+u_{il})\\ &= \left(\frac{b}{b_{max}}\right)^{n-\eta-1}(u_{agg}+u_{il})\end{aligned}$$

□

Theorem 2. *Given an IL proposer P with a utility u_{il} for inclusion of its input list and u_{agg} for winning the auction for aggregation. Given η input lists are available (except its own), $b_{max} > 2$ and the total number of IL proposers is n.*

1) *Given P generates a bias $b \leq 1$. Except with a negligible probability, making its input list available is the dominant action for P.*
2) *Given P generates a bias $1 < b < b_{max}-1$. Except with a negligible probability, making its input list available is the dominant action for P. Consequently (from parts 1 and 2 of the theorem), at least $\frac{b_{max}-1}{b_{max}}$ of the parties broadcast their input list in expectation.*
3) *Given P generates a bias $b \geq b_{max}-1$. The Nash equilibrium for the game would be a mixed strategy, i.e., make its input list available with some probability and withhold with some probability.*

Proof. 1) From Lemma 3, the utility from making its input list available is u_{il}. From Lemma 5, the utility from making its input list available is

$$\left(\frac{b+1}{b_{max}}\right)^{\eta}\cdot\left(\frac{b}{b_{max}}\right)^{n-\eta-1}(u_{agg}+u_{il})$$

Since $b_{max} > 2$ and $b \leq 1$, this utility tends to 0. Thus, the utility from making its input list available (u_{il}) is greater than that of not making its input list available (0). Thus, all such parties would make their input list available.

2) From Lemma 4, the utility from making its input list available is

$$u_{il}+\left(\frac{b}{b_{max}}\right)^{\eta}\cdot\left(\frac{b-1}{b_{max}}\right)^{n-\eta-1}u_{agg}$$

From Lemma 5, the utility of not making its input list available is

$$\left(\frac{b+1}{b_{max}}\right)^{\eta}\cdot\left(\frac{b}{b_{max}}\right)^{n-\eta-1}(u_{agg}+u_{il})$$

Consider the difference between the utilities.

$$\begin{aligned}&u_{il}+\left(\frac{b}{b_{max}}\right)^{\eta}\cdot\left(\frac{b-1}{b_{max}}\right)^{n-\eta-1}u_{agg}\\ &\quad-\left(\frac{b+1}{b_{max}}\right)^{\eta}\cdot\left(\frac{b}{b_{max}}\right)^{n-\eta-1}(u_{agg}+u_{il})\\ &\geq u_{il}\left(1-\left(\frac{b+1}{b_{max}}\right)^{n-1}\right)\\ &\quad+u_{agg}\left(\left(\frac{b}{b_{max}}\right)^{n-1}-\left(\frac{b+1}{b_{max}}\right)^{n-1}\right)\\ &\geq u_{il}\left(1-\left(\frac{b+1}{b_{max}}\right)^{n-1}\right)-u_{agg}\left(\frac{b+1}{b_{max}}\right)^{n-1}\\ &= u_{il}\left(1-\left(1+\frac{u_{agg}}{u_{il}}\right)\left(\frac{b+1}{b_{max}}\right)^{n-1}\right)\end{aligned}$$

To ensure this is ≥ 0, we require

$$\left(\frac{b+1}{b_{max}}\right)^{n-1} < \frac{u_{il}}{u_{il}+u_{agg}}$$

The probability for this is given by

$$\mathbb{P} = \frac{b_{max}\left(\frac{u_{il}}{u_{il}+u_{agg}}\right)^{1/n-1}-1}{b_{max}-1}$$

which is approximately 1 if n is large. Thus, parties with $b < b_{max}-1$ are incentivized to make their input list available. This occurs with a probability of $\frac{b_{max}-1}{b_{max}}$, which implies that in expectation, at least $\frac{b_{max}-1}{b_{max}}\cdot n$ parties would make their input lists available.

3) From Lemma 4, we know that the utility from making its input list available is

$$u_{il}+\left(\frac{b}{b_{max}}\right)^{\eta}\cdot\left(\frac{b-1}{b_{max}}\right)^{n-\eta-1}u_{agg}$$

From Lemma 6, the utility of not making its input list available is

$$\left(\frac{b}{b_{max}}\right)^{n-\eta-1}(u_{agg}+u_{il})$$

From Part 2, we know that η is in expectation more than $\frac{b_{max}-1}{b_{max}}n$. Substituting in Lemma 6, we get

$$\left(\frac{b}{b_{max}}\right)^{\frac{n}{b_{max}}-1}(u_{agg}+u_{il})$$

As b approaches b_{max}, the difference in utility is

$$\begin{aligned}&u_{il}+0-\left(1-\left(\frac{n}{b_{max}}-1\right)\left(\frac{b_{max}-b}{b_{max}}\right)\right)(u_{agg}+u_{il})\\ &=\left(\left(\frac{n}{b_{max}}-1\right)\left(\frac{b_{max}-b}{b_{max}}\right)\right)(u_{agg}+u_{il})-u_{agg}\end{aligned}$$

which is negative since the first term approaches 0.

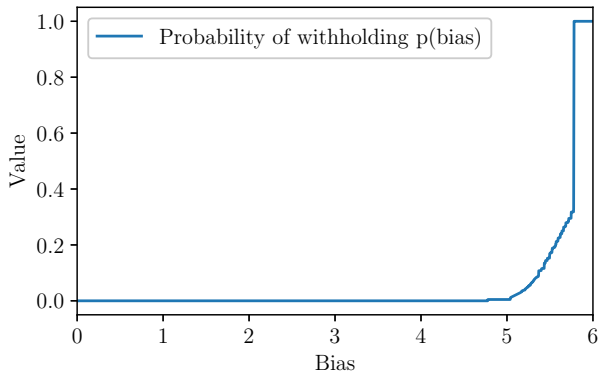


Figure 2: An example for the probability of withholding input list with the bias generated for IL Proposers. Example parameters: $n = 36, b_{max} = 6, u_{agg} = 6u_{il}$. In this example, the resulting value of γ turns out to be 0.946.

Thus, at least some parties are incentivized not to make their input list available, and a mixed Nash equilibrium follows.

The chart in Figure 2 shows the probability with which the IL Proposer would withhold its input list at equilibrium versus the bias drawn through the VRF (not make it available to other IL Proposers). □

6. Censorship Resistance

Combining the protocols in Sections 4 and 5, we get our complete end-to-end design for an inclusion list creation as follows:

- All IL proposers create an input list (InpL) each from a set of transactions as determined by Algorithm 1.
- All IL proposers create and broadcast inclusion lists trying to maximize the number of available input lists as described in Algorithm 3.

The above-described scheme works well in the absence of any external incentives. However, an adversary trying to censor a transaction can manipulate the incentives of the IL proposers with the following actions (for the rest of the paper, we would refer to the transaction the adversary wants to exclude as a *target* transaction m_t, and the utility IL proposers get from including the transaction is u_t, which is suggested to a set of parties N_t, where $|N_t| = n_t$):

1) Add transactions in the mempool to naturally manipulate the number of parties that are suggested to include m_t (n_t).
2) Bribe IL proposers to exclude the target transaction from the suggested input list.
3) Bribe IL proposers to exclude all input lists that contain the target transaction from the generated bid.
4) Exclude a bid containing at least one input list with the target transaction by crashing the party creating the bid.

In the following subsections, we will look at each of these actions. Initially, assume that Actions 1, 2 affect the input list building scheme and Actions 3, 4 affect the aggregation scheme. The two pairs of actions are exclusive and do not affect each other; later, after looking at the results, we will defend why this assumption is justified.

6.1. Censorship Resistance for Input-List building

We would first analyze Action 2 and then look at Action 1 and the interplay between the two. The first thing to note is that since some parties choose not to make their input lists available, this reduces the number of expected parties from the number of actual parties suggested to include the target transaction, the adversary does not know which parties would not broadcast. Thus, when considering the parties to bribe, the adversary would consider all parties in the set $|N_t|$ as opposed to the expected number of parties that broadcast ($N[s]$ in algorithm 1). However, we need to consider the expected number of broadcast input lists when considering the utility that the IL proposers receive from transactions. Let n_t be the number of input lists suggested to include the transaction ($n_t = |N_t|$), $\overline{n_t}$ as the expected number of broadcast input lists that contain the target transaction ($\overline{n_t} = n_t \cdot \gamma$, where γ is the probability that an IL proposer broadcasts), $\widetilde{n_t} = 1 + \gamma(n_t - 1)$ represents the denominator as specified in Eq (5) and $\widehat{n_t}$ represents the actual number of input lists that broadcast the transaction.

The expected utility for each IL proposer from including the target transaction is $\frac{u_t}{\widetilde{n_t}}$ and excluding it and adding another transaction m_s gives some utility $\frac{u_s}{\widetilde{n_s}+\gamma}$. If m_s is available in the global mempool, then $\frac{u_s}{\widetilde{n_s}+\gamma} \leq \frac{u_t}{\widetilde{n_t}}$, otherwise the cost u_s would be an additional cost to the adversary (and $n_s = 0$). Also, consider the utility for including the input list as suggested but without the target transaction to be u_{il}^-, which is the sum of utility that each transaction in the list provides.

Lemma 7. *Given a correlated equilibrium strategy as described in Algorithm 1, and a set N_t ($|N_t| = n_t$) of IL proposers who are suggested to include the target transaction m_t. Consider a bribe (br) from the adversary to IL proposers in the set N_t to censor the target transaction (action 2) and replace it with a replacement transaction m_s.*

i) If $br \leq \frac{u_t}{\widetilde{n_t}} - \frac{u_s}{\widetilde{n_s}+\gamma}$, the IL proposer would reject the bribe and propose the suggested input list.
ii) If $\frac{u_t}{\widetilde{n_t}} - \frac{u_s}{\widetilde{n_s}+\gamma} < br < u_t - \frac{u_s}{\widetilde{n_s}+\gamma}$, the IL proposers would reject the bribe with some non-zero probability.
iii) If $br \geq u_t - \frac{u_s}{\widetilde{n_s}+\gamma}$, the IL proposer would always accept the bribe and ignore the transaction.

Proof. The utility received by including the transaction is $\mathbb{U}_r = u_{il}^- + \frac{u_t}{\widehat{n_t}}$, where $\widehat{n_t} = \widetilde{n_t}$, if all IL proposers include the transactions as suggested. If some IL proposers accept the bribe and exclude the transaction from their input list, then $\widehat{n_t} < \widetilde{n_t}$. The utility received for accepting the bribe is $\mathbb{U}_a = u_{il}^- + \frac{u_s}{\widetilde{n_s}+\gamma} + br$. Thus, $\mathbb{U}_r - \mathbb{U}_a = \frac{u_t}{\widehat{n_t}} - \frac{u_s}{\widetilde{n_s}+\gamma} - br$. Since $1 \leq \widehat{n_t} \leq n_t$, we have that $\frac{u_t}{n_t} - \frac{u_s}{\widetilde{n_s}+\gamma} - br \leq \mathbb{U}_r - \mathbb{U}_a \leq u_t - \frac{u_s}{\widetilde{n_s}+\gamma} - br$. We now prove each part of the lemma separately.

i) If $br \leq \frac{u_t}{\widetilde{n_t}} - \frac{u_s}{\widetilde{n_s}+\gamma}$, $\mathbb{U}_r - \mathbb{U}_a \geq 0$. Since the utility gained by the IL proposer from accepting the bribe is less than rejecting the bribe, the IL proposer would always reject the bribe.

ii) If bribe $\frac{u_t}{\widetilde{n_t}} - \frac{u_s}{\widetilde{n_s}+\gamma} < br < u_t - \frac{u_s}{\widetilde{n_s}+\gamma}$, then the bribe is higher than the individual utility gained by the IL proposer if all other IL proposers choose to include the transaction (i.e., reject the bribe, $\widehat{n_t} = \widetilde{n_t}$). However, if all IL proposers choose to accept the bribe, then the utility received from rejecting the bribe and being the only IL proposer to include the target transaction is u_t. Thus, it is not rational for all IL proposers to accept or reject the bribe. A mixed Nash equilibrium would exist since both pure strategies are not an equilibrium, implying a non-zero probability of rejecting the bribe.

iii) If $br \geq u_t - \frac{u_s}{\widetilde{n_s}+\gamma}$, $\mathbb{U}_r - \mathbb{U}_a \leq 0$. Since the bribe available is greater than the utility that the IL proposer could receive, even if any other party does not include the transaction, the IL proposer always chooses to accept the bribe.

□

Lemma 8. *Given the correlated equilibrium strategy in Algorithm 1, the adversary must pay at least $u_t * (n_t - 1)$ to ensure (with probability = 1) that the target transaction does not appear in any input lists.*

Proof. From Lemma 7, if the adversary bribes the IL proposers in the set N_t in such a way that the IL proposers always censor the transaction, then the bribe has to be at least $u_t - \frac{u_s}{\widetilde{n_s}+\gamma}$. Now, if m_s is a public transaction, i.e., available in mempool, then $u_t - \frac{u_s}{\widetilde{n_s}+\gamma} \geq u_t - \frac{u_t}{\widetilde{n_t}} \geq u_t - \frac{u_t}{n_t}$. Since, at least n_t parties receive this bribe, the total cost to the adversary is at least $u_t * (n_t - 1)$ to bribe all the IL proposers in set N_t. However, if the transaction is private, then the cost to the adversary is $u_s + br = u_t$. This amount must be given to all n_t parties, and thus the total cost would be $u_t n_t > u_t * (n_t - 1)$ □

Thus, if the adversary chooses a strategy to bribe IL proposers to exclude the target transaction (Action 2), it must pay a cost of at least $u_t * (n_t - 1)$.

Consider Action 1. The adversary may choose to add spam transactions in such a way that the target transaction does not appear in any input list, or it may choose to reduce the number of n_t, and then follow up with Action 2.

Before we proceed with the analysis of this action, we need to observe the dependence of n_t on the fee paid by the transaction u_t and the fee paid by other transactions. Let $T(M)$ represent all the transactions the input list-building mechanism chooses. From Eq (5) for a correlated equilibrium, we know that,

$$\forall m_i \in T(M) : \frac{u_i}{\widetilde{n_i}} \geq \frac{u_t}{\widetilde{n_t} + \gamma}$$

Since $\widetilde{n_i} > 0$ for all $m_i \in T(M)$, we have

$$\forall m_i \in T(M) : u_i(\widetilde{n_t} + \gamma) \geq u_t \widetilde{n_i}$$

Taking the sum over all $m_i \in T(M)$, and noting that $\frac{u_t}{\widetilde{n_t}} > \frac{u_t}{\widetilde{n_t}+\gamma}$

$$\sum_{i \in T(M)} u_i(\widetilde{n_t} + \gamma) > u_t \sum_{i \in T(M)} \widetilde{n_i}$$

$$\sum_{i \in T(M)} u_i(\widetilde{n_t} + \gamma) > u_t \cdot n \cdot k \cdot \gamma$$

Let σ represent $\sum_{i \in T(M)} u_i$. Also, $\widetilde{n_t} = 1 + \gamma(n_t - 1)$. Thus,

$$1 + \gamma n_t > \gamma \cdot n \cdot k \frac{u_t}{\sigma}$$

$$n_t > n \cdot k \frac{u_t}{\sigma} - \frac{1}{\gamma} \tag{9}$$

Using Action 1, the adversary can censor the target transaction by adding transactions to the mempool. If more transactions are to be chosen, then the algorithm would suggest the transaction to fewer parties. If we view these extra transactions as a sequential addition of transactions to the mempool, we will reach a point where the target transaction is suggested to only one IL proposer. In other words, the following lemma compares using Action 1 to censor completely and a hybrid of Action 1 and Action 2 to first reduce the number of IL proposers and then bribe the rest.

Lemma 9. *Given the correlated equilibrium strategy in Algorithm 1, if the adversary reduces the number of times the transaction is suggested to $n_t = 1$ (Action 1), then after reaching this state, the cost incurred to an adversary in bribing the IL proposer (Action 2) is lower than adding further transactions to reduce the number of parties that are suggested to include the transaction (Action 1).*

Proof. To displace the target transaction, the adversary would have to displace all selections of the transactions after the target transaction is chosen for the first time in Algorithm 1 since each of the transactions that are selected after the target transaction gave lower utility than the first selection of the target transaction. Thus, in the worst case for $n_t = 1$, the target transaction is the last transaction chosen in Step 1 of Algorithm 1 such that there are no other transactions to displace. To displace the last chosen object (which in this case is m_t), the adversary would need to add a transaction m_a such that $\frac{u_a}{\widetilde{n_a}+\gamma} \geq \frac{u_t}{\widetilde{n_t}}$. Since $n_t = 1$, $\widetilde{n_t} = 1$, this implies $u_a \geq u_t$. Thus, Action 1 costs at least u_t.

If the adversary instead chooses to bribe the IL proposer, then from Lemma 7 the minimum bribe it would have to pay the IL proposers is $u_t - u_s \leq u_t \leq u_a$, where u_s represents the utility of some replacement transaction that the IL proposer could include. Thus, Action 2 costs at most u_t.

Thus, bribing the IL proposer dominates the action of displacing the transaction through added adversarial transactions when the target transaction appears only once. □

The above lemma proves that a hybrid of Action 1 and Action 2, reducing the number of times the target transaction

appears by spamming and then bribing the remaining IL Proposers that are suggested to include the transaction is a dominant action over Action 1, spamming transactions to remove the target transaction from inclusion. The next lemma compares the hybrid of Action 1 and Action 2 with only Action 2.

Lemma 10. *Given the correlated equilibrium strategy in Algorithm 1, if the adversary adds adversarial transactions (Action 1) with a total fee of u_a and pays a bribe br_1 to all the IL proposers which are suggested to add the target transaction (Action 2). If $u_t \leq \frac{\sigma}{\sqrt{nk}}$, then the cost incurred by the adversary is greater than $u_t(n \cdot k\frac{u_t}{\sigma} - 1 - \gamma)$*

Proof. If the adversary does not add any adversarial transaction, then the cost to the adversary by only bribing is given from Lemma 8 and Eq.(9). This cost is represented by

$$C = u_t(n \cdot k\frac{u_t}{\sigma} - 1 - \frac{1}{\gamma})$$

Since no additional transactions are added, the cost to the adversary is only the bribe. Thus, $br_1 + u_a \geq C$ in this case.

If the adversary adds some transactions to reduce the number of times the target transaction appears in algorithm 1, and then censors the rest (hybrid of action 1 and action 2) then the cost to the adversary is given by the fees paid plus the bribe cost to remove the target transaction from the reduced number of input lists.

$$C_1 = u_a + br_1$$

From Lemma 8 and Eq.(9), we have

$$\begin{aligned} br_1 &\geq u_t({n_t}' - 1) \\ {n_t}' &\geq n \cdot k \cdot \frac{u_t}{\sigma - \sum(u_l) + u_a} - \frac{1}{\gamma} \\ &\geq n \cdot k \cdot \frac{u_t}{\sigma + u_a} - \frac{1}{\gamma} \end{aligned}$$

, where $\sum(u_l)$ represents the sum of any transactions removed. Thus,

$$C_1 \geq u_a + u_t(n \cdot k \cdot \frac{u_t}{\sigma + u_a} - 1 - \frac{1}{\gamma})$$

The difference in this cost to the adversary and the minimum cost we claim is given by

$$\begin{aligned} C_1 - C &\geq u_a - u_t(nku_t) \cdot \left(\frac{1}{\sigma} - \frac{1}{\sigma + u_a}\right) \\ &\geq u_a - nku_t^2 \left(\frac{u_a}{\sigma(\sigma + u_a)}\right) \end{aligned}$$

If $u_t \leq \frac{\sigma}{\sqrt{nk}}$, then $u_t^2 \leq \frac{\sigma^2}{nk}$. Thus,

$$C_1 - C \geq u_a - \sigma^2 \left(\frac{u_a}{\sigma(\sigma + u_a)}\right) \geq 0$$

Thus, $C_1 \geq C$ and the minimum cost that the adversary must pay is $u_t(n \cdot k\frac{u_t}{\sigma} - 1 - \frac{1}{\gamma})$.

□

6.2. Censorship Resistance for Aggregation Step

The next set of actions that an attacker can take to censor a transaction is to target the aggregation algorithm and ensure the aggregated list does not contain the target transaction in any of the input lists. We first give the adversary the advantage that any input list that is broadcast but not made available (i.e., **F** is set to 0), then the cost to remove such a list from the bid is 0. Let $\widehat{n_t}$ be the number of input lists made available with the target transaction, m_t. This could be less than the number of parties suggested to include the target transaction in the input list due to the effect of Actions 1 and 2 by the adversary. To censor the transaction in Algorithm 3, the adversary must ensure that the highest bid selected by the block proposer excludes all the input lists containing the target transaction. Let's parameterize the blockchain's requirement to include a proof that the bid is greater than $n - \theta$ other bids. We give the adversary absolute control over which bids are dropped due to threshold requirements. In other words, the adversary must ensure that at least one of the bids within the top $\theta + 1$ bids does not contain any input lists with the target transaction.

Verifiable Random Functions (VRFs) guarantee the privacy of the bias generated by each IL proposer. Thus, we will assume that the adversary would not know the bias generated by the IL proposer. This does not prevent the adversary from bribing the IL proposer to get this information.

The first thing to note here is that the adversary can infer from Theorem 2 that if a proposer has not made its input list available, then the bias for such a party must be larger than one less than the maximum bias, i.e., $b > b_{max} - 1$. However, it cannot tell that if an IL proposer made its input list available that the bias for the party is $\leq b_{max} - 1$, since an IL proposer may still choose to make its input list available even if the bias for it is $> b_{max} - 1$ (since it is a mixed Nash equilibrium). Next, we also note that Action 4 cannot censor the target transaction since, under honest conditions, each bid would contain all input lists, including those containing the target transaction; excluding θ of them would not censor the target. Thus, we would look at Action 4 as a sub-routine within Action 3.

Lemma 11. *Given $\widehat{n_t}$ is the number of input lists that contain the target transaction, m_t. Given an IL proposer P which generates a bias $b \geq b_{max} - \widehat{n_t}$. If $br < u_{agg}/2$, then P would reject the bribe with some non-zero probability. If $br \geq u_{agg}/2$, then P would accept the bribe.*

Proof. In order to remove the target transaction m_t, the adversary requires the IL proposers to exclude all $\widehat{n_t}$ input lists that contain it. This would reduce the bid the IL proposer can send by $\widehat{n_t}$. Let's consider the case where $br \leq u_{agg}/2$. At equilibrium, let the probability with which the bribe is rejected be p. Consider the case of $p = 0$. If all IL proposers decide to accept the bribe, then if P rejects the bribe, the adversary would drop its bid amongst the θ crash faults tolerated. However, in subsequent blocks, this bid would be included with proof that the bid was higher

than the winning bid. (There is no incentive for the adversary to censor it in later rounds). This yields a utility of $u_{agg}/2$ for P. Thus, the incentive from rejecting the bribe is at least $u_{agg}/2$. If the bribe is less than $u_{agg}/2$, then all parties would have an incentive to reject the bribe with some non-zero probability.

For the case that $br \geq u_{agg}/2$, if the IL proposer rejects the bribe, the maximum utility it can get is by winning the highest bid reward of $u_{agg}/2$ (while it would also have to pay a fee to get its bid included in the later round). Thus, it would always accept the bribe if $br \geq u_{agg}/2$. □

As a consequence of Lemma 11, if the bribe offered is $< u_{agg}/2$, then the number of bids submitted by IL proposers that do not accept bribes is (with some probability) greater than θ. Thus, the bribery fails with some probability.

Lemma 12. *Given $\widehat{n_t}$ is the number of input lists that contain the target transaction, and η is the total number of input lists available. If an adversary wants to censor the target transaction (with 100% probability) by bribing during the aggregation phase, then the total cost incurred by the adversary is at least $(n-\theta) \cdot u_{agg}/2$.*

Proof. From Lemma 11, the minimum bribe required to bribe an IL proposer who draws a bias $> b_{max} - \widehat{n_t}$ would be $u_{agg}/2$. However, the adversary does not know which parties draw such a bias. The adversary can identify that each IL proposer that did not broadcast the input list would have (with a high probability) a bias greater than $b_{max} - 1$; however, this does not give any information about an IL proposer drawing a bias less than $b_{max} - n_t$. This implies that the adversary would have to bribe all but θ IL proposers regardless of the value of bias drawn. Thus, the total bribe the adversary has to pay is $(n-\theta) \cdot u_{agg}/2$. □

From Lemma 12, we observe that any bribery for parties in the aggregation phase (hybrids of Actions 3 and 4 is independent of the number of times the target transaction appears in the input lists. Thus, a reduction of the number of times the target transaction appears in input lists by Actions 1 and 2 has no reduction in the cost to an adversary when it takes Actions 3 and 4. Thus, the two sets of actions are independent.

Consider the following parameterization of the protocol. $b_{max} = \sqrt{n}, u_{agg} = \sqrt{n} * u_{il}$. The sum of rewards for the input list across all parties is the same as the sum of fees paid by all transactions in the inclusion list, i.e., $\sum_{j \in N} u_{il}^j = \sum_{i \in T(M)} f_i = \sigma$. And thus, the expected u_{il} for each party is σ/n. Also, by Theorem 4 in Appendix D, the reward distribution is roughly the same across all parties, and thus, we can say that the reward for each party does not deviate from the expected reward by much. For this protocol, we claim the following:

Theorem 3. *Given n parties running the protocol, M represents the transactions available to all parties in the mempool, f_j represents the fee paid by a transaction $m_j \in M$, $\theta - 1$ represent the number of crash faults tolerated, and T represent the union of all lists L_j when Algorithm 1 is run on M. Consider $B = \{br_1, \ldots, br_{|T(M)|}\}$ such that $br_i = \max((n \cdot k\frac{f_j}{\sigma} - 1 - \frac{1}{\gamma})f_j, \frac{(n-\theta)}{\sqrt{n}}\sigma)$. The protocol satisfies (B, θ, T)-censorship resistance.*

Proof. Consider an adversary with a bribery budget of br. From Lemma 10, we know that if the adversary attempts to censor a transaction m_j from the input list, the least amount of bribe it must pay is

$$(n \cdot k\frac{f_j}{\sigma} - 1 - \frac{1}{\gamma})f_j$$

Note that here, $kf_j < \sigma$ and $f_j \leq \frac{\sigma}{\sqrt{nk}}$ If the adversary attempts to censor the transaction in the aggregation phase, then the total cost, as governed by Lemma 12 is

$$\begin{aligned}(n-\theta)u_{agg} = (n-\theta)\sqrt{n}u_{il} &\geq (n-\theta)\sqrt{n}\sigma/n \\ &\geq \frac{(n-\theta)}{\sqrt{n}}\sigma\end{aligned}$$

Now, let $br_i = \max((n \cdot k\frac{f_j}{\sigma} - 1 - \frac{1}{\gamma})f_j, \frac{(n-\theta)}{\sqrt{n}}\sigma)$.

If $br < br_i$, then at least θ parties will output the inclusion list, which includes the transaction m_i, implying that the proposer will select the inclusion list with transaction m_i at least once. Thus, the protocol is (B, θ, T)–censorship resistant. □

7. Discussion

On unconditional inclusion lists. An essential property for inclusion lists we consider is that all transactions in the list must be included in the next block. This limits the maximum size of the inclusion list to be less than the block size. Since each party can choose to propose transactions with no overlap, the maximum size of the input list needs to be restricted to the size of the block divided by the number of parties in the system. We consider the notion of unconditional inclusion lists since if there is a way for a proposer to exclude the transactions in the inclusion list, then it creates a single point of failure that the adversary can exploit. If the number of transactions available in the inclusion list is greater than the size of the block, then in such cases, the cost to censor the transaction is just the difference in the fee paid by the target transaction to the proposer and that for its replacement (which may not be from the IL).

Inclusion lists with EIP 1559. In Ethereum, due to the presence of the EIP-1559 fee mechanism [15], in expectation, block sizes fluctuate around half of the limit. This counters the previous discussion point since, in most cases, there would be enough space for transactions in the block. If the adversary adds spam transactions to fill the leftover block space, the adversary will incur an additional cost corresponding to the transaction fee (base fee in EIP 1559) of half the block size limit. However, in doing so, the adversary would also increase the base fee for the next block, which may lead to censorship by raising the base

fee above the fee. In the absence of these, the input list-building algorithm used could be replaced with a much simpler rule to include all transactions (or capped to block limit to prevent spamming like in [14]). This does not affect the second part of the design - AUCIL, where inclusion lists are formed by aggregation of input lists. Compared to prior designs such as FOCIL [13], in AUCIL, we do not rely on the honesty of the attestors to collect and aggregate the list locally. Compared to FOCIL, where attersters need to receive all local inclusion lists and compute a running aggregate locally, AUCIL only requires a simple verification of proof. If they do not correctly verify, they could be slashed for incorrectly voting on the progress of the block. In an alternate design where fraud-proofs [17] is used, the proposer who adds the incorrect proof would be slashed. In this case, the attestors are not involved in verifying the validity of the proof either.

Common mempool assumption. In our protocol for input list building algorithm 1, we assume that all parties have the same view of transactions. We note that transactions that pay a fee for obtaining censorship resistance guarantees would necessarily be transmitted through public channels (as opposed to transactions sent as private order flows to only some providers). Hence, any transaction received by one party will be received by all parties soon enough. There may still be minor differences in the mempool of parties due to the time for transmitting a transaction to others. Accounting for these differences in the protocol design is an important future work.

Practical considerations. Compared to other inclusion list designs [12], [13], AUCIL has a lower overhead largely in part due to the restricted size of data that each party has to share. However, being a 2-step protocol where first input lists are broadcast and then the inclusion lists are created as bids, the number of communication rounds increases. Extrapolating numbers from EIP-7805 [14] for inclusion lists in Ethereum, the limits on the size of the input list could be set to $k = 6$ average-sized transactions (or 3 kB of data), and the number of parties in the committee is $n = 20$. This would, on average, imply an inclusion list containing 120 transactions.

Since we want to be robust against bribery, we introduce a new fee for each transaction and a reward for the aggregation of lists. Such a fee can replace the tip paid by the user in case an inclusion list route is taken or could be introduced in addition to the tip paid. Similarly the aggregation reward is currently treated as an out-of-protocol reward, i.e., the protocol would be generating rewards. However, this could be replaced with a contribution score mechanism as well, and the reward for aggregation could be extracted from the fee paid by the user. This design is left as future work.

Interpreting censorship resistance of our protocol. Consider Example 1 with $M = \{m_1, m_2, m_3,\ m_4, m_5, m_6\}$ paying a fee of $\{10, 9, 8, 3, 2, 1\}$. Both $T_1(M) = \{m_1, m_2, m_3, m_4\}$ and $T_2(M) = \{m_1, m_2, m_3\}$ are meaningful inclusion lists. The first one offers censorship resistance to more transactions. But the second one resists a bribery budget of $B_2 = \{20, 9, 8\}$ as compared to $B_1 = \{10, 9, 8, 3\}$; thus, higher paying transactions have more resistance to censorship. Another example could have been $T_3(M) = \{m_1\}$ with $B_3 = \{40\}$ and offers an even higher resistance to censorship. Our inclusion list outputs $T_2(M)$ even when the maximum possible size of the inclusion list was 4. There could have been higher censorship resistance for one transaction (T_3), or censorship resistance could have been provided to more transactions (T_1). However, $T_2(M)$ is the only equilibrium among the three allocations, and unless an equilibrium allocation is considered, the analysis of censorship resistance in the allocation is not fruitful. IL Proposers would switch from the allocation to one with higher utility.

Commit reveal scheme with a mixed Nash Equilibrium for input list building. As an alternative to using a correlated equilibrium scheme as described in algorithm 1, we could use a commit reveal scheme that avoids the timing games. This potentially could lead to better censorship resistance. However, one of the major properties of the inclusion list-building scheme is that no consensus needs to be reached, and without consensus, committing to a bid before revealing it is infeasible.

Secret Ordering of IL proposers In Section 6 for algorithm 1, we assume that the adversary knows the ordering of IL proposers, i.e., if it simulates the algorithm, it would know exactly which party would include the target transaction in its input list. However, in a design where this information is unknown to the adversary, it would have to bribe all the parties to ensure that none of the parties include the target transaction. Making this order secret has its challenges. All IL proposers would need to know their exact position but not have any information about the position of others IL proposers to avoid single-party bribery revealing the entire sequence. This does not have to be a verifiable method since no party has incentives to switch its position (due to the proven correlated equilibrium). A secret ordering mechanism would thus ensure the theoretically maximum censorship resistance (linear in number of parties) while also ensuring the maximum utility for each party.

8. Related Work

Bribery-based censorship attacks. Blockchains have been known to be vulnerable to bribery-based censorship attacks [7], [19], [20]. These attacks have been known to affect the security of various applications like AMMs [21], [22], atomic swaps [7], [23], [24], [25] and auctions [8]. Censorship resistance was formally studied in [8], where they show the extent of the problem modeling censorship into the consensus can bring for financial applications like an auction.

Inclusion list designs. To our knowledge, this paper is the first to introduce an inclusion list formally designed to combat censorship in the literature. However, there are some

ideas presented in research posts of some blockchains [10], [12], [13], [26].

In forward Inclusion List designs [10], an inclusion list is published by the previous block's proposer. All transactions in the list must be considered in the next block if block space remains underutilized. This forces builders to fill blocks, making censorship more costly, as any unused block space must be used to include proposer-specified transactions. However, owing to a single proposer-based proposal, this faces major bribery-based censorship issues in which an adversary could bribe the proposer to remove the transaction from the inclusion list itself.

Multi-party designs like COMIS [12] and FOCIL [13] address the low cost of bribery by relying on a committee to create the inclusion list. Intuitively, they argue that if more parties include the transaction in their inclusion lists, the amount of bribes the adversary pays increases. A practical version of this design-FOCIL-has been pushed as EIP-7805 [14] which limits the size of each input; however, the total size limit is still greater than the capacity of the block and could thus overflow. However, such designs fail to account for how these committee members will create their inputs to the inclusion list - with a basic assumption that highest paying transactions would be chosen - and, thus, fail to provide guarantees when the network is busy, i.e., there are enough transactions to fill the block.

In Flashbots report [26], inclusion lists protocols are studied for censorship resistance in block-chains, albeit under an honest and Byzantine model, where a known threshold of parties can be Byzantine, and the rest are considered honest. The definition of censorship resistance used differs from our definition. They consider the time required to include a transaction in the chain as a parameter for censorship and try to reduce it with various protocols. They analyze leader-based protocols with inclusion lists and note that using a data availability layer or reliable broadcast can help reduce censorship in their design. While they mention the number of parties that need to be bribed in a world where all parties are rational, they do not show what amount of bribes is required or a formal analysis of why the number of bribes cannot be reduced.

Multi-proposer based designs. In [8], in addition to modeling bribery in auctions, they propose mitigating censorship by adding multiple proposers to produce a block simultaneously. While doing so, they propose a dual fee structure in which if the transaction is proposed only once, the tip paid to the party that includes it is higher. This would increase the cost of censorship for the adversary to be proportional to the higher tip (which is rarely paid by the user) instead of the general tip. However, in doing so, they inadvertently prioritize solo inclusion of transactions, reducing the number of times the transaction would be repeated in case more transactions are pending than the size of the block and not all proposers can add all transactions in their local block.

In Flashbots report [26], they also introduce a multi-proposer system called Partially Ordered Dataset (POD) to partially order the available set of transactions. Unlike traditional consensus mechanisms that impose a strict transaction order, POD assigns timestamps that loosely order transactions across replicas. Transactions can be submitted to any replica. They will be recorded as long as they reach a quorum of honest replicas, making it difficult for any single entity to block or censor them entirely. Additionally, POD includes mechanisms for detecting and documenting censorship attempts, creating accountability for malicious behavior. By supporting high throughput and rapid transaction propagation, POD reduces the window in which censorship could occur. A formal analysis in the presence of rational parties is not provided.

References

[1] M. Holden, "WikiLeaks says "blockade" threatens its existence | Reuters." [Online]. Available: https://www.reuters.com/article/us-britain-wikileaks/wikileaks-says-blockade-threatens-its-existence-idUSTRE79N46K20111024/?feedType=RSS&feedName=topNews

[2] "Canada convoy protest," Sep. 2024, page Version ID: 1243947900. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Canada_convoy_protest&oldid=1243947900

[3] Robinhood, "Keeping Customers Informed Through Market Volatility," Jan. 2021. [Online]. Available: https://newsroom.aboutrobinhood.com/keeping-customers-informed-through-market-volatility/

[4] A. Robertson, "Robinhood is facing dozens of lawsuits over GameStop stock freeze," Feb. 2021. [Online]. Available: https://www.theverge.com/2021/2/1/22254656/robinhood-gamestop-stonks-trade-freeze-class-action-lawsuits

[5] U.S. Department of the Treasury, "OFAC Specially Designated Nationals Data," 2022. [Online]. Available: https://www.treasury.gov/ofac/downloads

[6] A. Wahrstätter, J. Ernstberger, A. Yaish, L. Zhou, K. Qin, T. Tsuchiya, S. Steinhorst, D. Svetinovic, N. Christin, M. Barczentewicz, and A. Gervais, "Blockchain Censorship," Jun. 2023, arXiv:2305.18545 [cs]. [Online]. Available: http://arxiv.org/abs/2305.18545

[7] F. Winzer, B. Herd, and S. Faust, "Temporary censorship attacks in the presence of rational miners," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2019, pp. 357–366.

[8] E. Fox, M. Pai, and M. Resnick, "Censorship resistance in on-chain auctions," 2023. [Online]. Available: https://arxiv.org/abs/2301.13321

[9] V. Buterin, "State of research: increasing censorship resistance of transactions under proposer/builder separation (pbs)," 2021. [Online]. Available: https://notes.ethereum.org/s3JToeApTx6CKLJt8AbhFQ#Hybrid-PBS-can-we-use-proposers-only-for-inclusion-of-last-resort

[10] Francesco. [Online]. Available: https://notes.ethereum.org/@fradamt/forward-inclusion-lists

[11] Michael, Vitalik, Francesco, Terence, potuz, and Manav, "Eip-7547: Inclusion lists," Oct 2023. [Online]. Available: https://eips.ethereum.org/EIPS/eip-7547

[12] Thomas, Francesco, and Barnabe, "The more, the less censored: Introducing committee-enforced inclusion sets (comis) on ethereum," Feb. 2024. [Online]. Available: https://ethresear.ch/t/the-more-the-less-censored-introducing-committee-enforced-inclusion-sets-comis-on-ethereum/18835

[13] "Fork choice enforced inclusion lists," Jun. 2024. [Online]. Available: https://ethresear.ch/t/fork-choice-enforced-inclusion-lists-focil-a-simple-committee-based-inclusion-list-proposal/19870

[14] E. I. Proposals, "Eip-7805: Fork-choice enforced inclusion lists (focil) [draft]," Nov. 2024. [Online]. Available: https://eips.ethereum.org/EIPS/eip-7805

[15] Vitalik, Eric, Rick, Matthew, Ian, and Abdelhamid, "Eip-1559," Apr 2019. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1559

[16] "Unconditional inclusion lists," Jan. 2024. [Online]. Available: https://ethresear.ch/t/unconditional-inclusion-lists/18500

[17] M. Al-Bassam, A. Sonnino, and V. Buterin, "Fraud proofs: Maximising light client security and scaling blockchains with dishonest majorities," *arXiv preprint arXiv:1809.09044*, vol. 160, 2018.

[18] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.

[19] J. Bonneau, "Why buy when you can rent? bribery attacks on bitcoin-style consensus," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 19–26.

[20] P. McCorry, A. Hicks, and S. Meiklejohn, "Smart contracts for bribing miners," in *Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers 22*. Springer, 2019, pp. 3–18.

[21] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges," *arXiv preprint arXiv:1904.05234*, 2019.

[22] C. F. Torres, R. Camino, and R. State, "Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 1343–1359. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/torres

[23] S. Wadhwa, J. Stoeter, F. Zhang, and K. Nayak, "He-htlc: Revisiting incentives in htlc," *Cryptology ePrint Archive*, 2022.

[24] H. Chung, E. Masserova, E. Shi, and S. A. Thyagarajan, "Rapidash: Foundations of side-contract-resilient fair exchange," *Cryptology ePrint Archive*, 2022.

[25] I. Tsabary, M. Yechieli, A. Manuskin, and I. Eyal, "Mad-htlc: because htlc is crazy-cheap to attack," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1230–1248.

[26] O. Alpos, B. David, N. Kamarinakis, and D. Zindros, "Flashbots report: System requirements, existing and new solutions, and their efficiency."

[27] E. Budish, "The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes," *Journal of Political Economy*, vol. 119, no. 6, pp. 1061–1103, 2011.

[28] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang, "The unreasonable fairness of maximum nash welfare," *ACM Transactions on Economics and Computation (TEAC)*, vol. 7, no. 3, pp. 1–32, 2019.

Appendix A. Nash Equilibrium

To compute the Nash Equilibrium for the input list-building scheme (with $\gamma = 1$), consider the following:

Let p_i represent the probability of choosing object m_i in a Nash equilibrium. In a Nash equilibrium, all parties follow the same probability distribution of choosing the object. Given this information, let party n_j be the decision-making party when picking objects and try to deviate from the given Nash Equilibrium Probability. For this party, let x_{p_i} represent the probability of choosing an object m_i. The utility for such a party is given by

$$\begin{aligned} u_j &= \sum_{i=1}^{m} x_{p_i} f_i \sum_{k=0}^{n-1} \frac{\binom{n-1}{k}(p_i)^k(1-p_i)^{n-k-1}}{k+1} \\ &= \sum_{i=1}^{m} x_{p_i} f_i (1-p_i)^{n-1} \sum_{k=0}^{n-1} \frac{1}{k+1}\binom{n-1}{k}\left(\frac{p_i}{1-p_i}\right)^k \end{aligned}$$

Let $c = \frac{p_i}{1-p_i}$

$$\begin{aligned} u_j &= \sum_{i=1}^{m} x_{p_i} f_i (1-p_i)^{n-1} \sum_{k=0}^{n-1} \binom{n-1}{k} \frac{c^k}{k+1} \\ &= \sum_{i=1}^{m} x_{p_i} f_i (1-p_i)^{n-1} \cdot \frac{1}{c} \cdot \sum_{k=0}^{n-1} \binom{n-1}{k} \frac{c^{k+1}}{k+1} \end{aligned}$$

Now, from the integral of binomial expansion of $(1+c)^{n-1}$(by parts from 0 to c)

$$\sum_{k=0}^{n-1} \binom{n-1}{k} \frac{c^{k+1}}{k+1} = \frac{(1+c)^n - 1}{n}$$

From this and substituting $c = \frac{p_i}{(1-p_i)}$,

$$\begin{aligned} u_j &= \sum_{i=1}^{m} x_{p_i} f_i (1-p_i)^{n-1} \cdot \frac{1}{c} \cdot \left(\frac{(1+c)^n - 1}{n}\right) \\ &= \sum_{i=1}^{m} \frac{x_{p_i}}{n \cdot p_i} f_i (1-p_i)^n \cdot \left((\frac{1}{1-p_i})^n - 1\right) \\ &= \sum_{i=1}^{m} \frac{x_{p_i}}{n \cdot p_i} f_i \cdot (1 - (1-p_i)^n) \end{aligned}$$

We can ignore the higher order term for any $0 < p_i < 1$. Thus,

$$u_j = \sum_{i=1}^{m} \frac{x_{p_i}}{n p_i} f_i$$

For the Nash equilibrium to exist at values $x_{p_i} = p_i, \forall i \in m$, all terms should be individually equal (so that multiplying by x_{p_i} yields the same value across all m terms). Thus, $\frac{f_i}{p_i} = \frac{f_j}{p_j}$. Since each party selects k objects, we have $\sum p_i = k$, this gives us

$$p_i = \frac{k f_i}{\sum_i f_i} \tag{10}$$

However, some p_i can also be 0. For this to be the case, we require that the utility term corresponding to i should be less than the utility of all other utility terms. Thus, $f_i \leq \frac{f_j}{n p_j}$ for all j such that $p_j \neq 0$. This implies if $f_i \leq \frac{\sum f_j}{nk}$ then $p_i = 0$.

Some p_i can exceed 1 as well, given the calculation needed to find the value. In such a case, the probability will be capped at 1, and all parties will choose that object.

The expected utility for the input list proposer would be less than or equal to $\frac{\sum_{i:p_i \neq 0} f_i}{n}$(in cases where due to randomness, one or more m_i with $p_i > 0$ did not get selected). As an example, if there exist two objects with values 3 each, and two parties select one object each. In this case, if both parties select the same object (occurs with probability 0.5), the utility for both parties is 1.5. In this case, the expected utility for both parties is 2.25, which is < 3.

Appendix B. Allocation Examples

To look at various allocations, consider the following example: 15 transactions exist with utilities 15, 14, 13, 11, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1. Let there be 5 IL proposers, choosing three transactions each.

Let's consider 5 different allocations as shown in Table 2. A1, A2, A3, and A4 represent the correlated equilibrium at various

	Party	Index	Utility			
			U_{100}	U_{70}	U_{50}	U_0
A0	Party 1	[0, 5, 10]	21	21	21	21
	Party 2	[1, 6, 11]	19	19	19	19
	Party 3	[2, 7, 12]	17	17	17	17
	Party 4	[3, 8, 13]	14	14	14	14
	Party 5	[4, 9, 14]	8	8	8	8
A1	Party 1	[1, 4, 6]	14.6667	15.8333	17	24
	Party 2	[0, 1, 3]	13.9167	17.1426	20.3333	40
	Party 3	[0, 2, 3]	13.5833	16.726	19.8333	39
	Party 4	[0, 2, 5]	13.0833	15.2554	17.5	33
	Party 5	[0, 1, 2]	12.75	16.0887	19.5	42
A2	Party 1	[0, 2, 4]	14.0833	16.2554	18.5	34
	Party 2	[0, 1, 2]	12.75	16.0887	19.5	42
	Party 3	[1, 3, 5]	13.3333	15.4167	17.5	30
	Party 4	[0, 1, 3]	12.0833	15.2554	18.5	40
	Party 5	[0, 2, 3]	11.75	14.8387	18	39
A3	Party 1	[0, 1, 2]	11.5833	14.7715	18.1	42
	Party 2	[0, 1, 2]	11.5833	14.7715	18.1	42
	Party 3	[2, 3, 4]	14	16	18	30
	Party 4	[0, 1, 3]	10.9167	13.9382	17.1	40
	Party 5	[0, 1, 3]	10.9167	13.9382	17.1	40
A4	Party 1	[0, 1, 2]	8.4	11.0526	14	42
	Party 2	[0, 1, 2]	8.4	11.0526	14	42
	Party 3	[0, 1, 2]	8.4	11.0526	14	42
	Party 4	[0, 1, 2]	8.4	11.0526	14	42
	Party 5	[0, 1, 2]	8.4	11.0526	14	42

TABLE 2: Utility Received by Each Party for Different Allocations. U_p represents the utility of broadcasting the list if the probability of broadcasting is $p\%$

Appendix C. Example for Algorithm 1

This example demonstrates the operation of algorithm 1 with $n = 3$ parties, $m = 5$ objects, $k = 2$ size of inclusion list and $U = [8, 6, 5, 3, 1]$ utility values of the objects.

In Step 1, the algorithm iteratively selects the highest-value objects from U, dynamically adjusting selection counts in N. This process continues until $n \cdot k$ selections are made or all objects are fully allocated.

In Step 2, the algorithm distributes the selected objects among the players. The objects are allocated based on their adjusted utilities U_f and assigned in decreasing order of U_f. Each round assigns objects in order, updating the players' inclusion arrays L_i. Step 2 starts with $U = [8, 6, 5, 3, 1$, $N = [2, 2, 1, 1, 0]$, $U_f = [4, 3, 5, 3, 0]$ and $A = [2, 0, 1, 3, 4]$

Appendix D. Fairness for Algorithm 1

Randomness has its limitations. On-chain randomness is prone to grinding attacks and thus is not enough to guarantee that the rewards from the input list-building scheme are equally distributed. Thus, we prove that the round-robin allocation of objects through algorithm 1 with $\gamma = 1$ follows multiple definitions of fairness known in the literature. The definition as proposed by [27]

Lemma 13. *(EF1-Fairness) The allocation achieved in algorithm 1 is 1-Envy Free.*

Proof. The proof for the algorithm is simple. During Step 2, objects are allocated in a round-robin after sorting. Let o_i^r be allocated to party i in round r. The following two properties hold due to sorted allocation.

$$\mathbb{U}(o_i^r) \leq \mathbb{U}(o_j^r) \ \forall r; i < j$$
$$\mathbb{U}(o_i^r) \leq \mathbb{U}(o_j^k) \ \forall j; k < r$$
$$\text{where, } \mathbb{U}(o_i^r) = \frac{u_{o_i^r}}{n_{o_i^r}}$$

Let party j envy party i. If $j < i$, then in each round, the utility gained by party j is greater than party i, and thus, there is no envy. For $j > i$, let's remove the first object party i received. Thus, for each object received by party i in round r, there exists an object in round $r - 1$ that party j receives. Since the utility gained from an earlier round is always more than that gained in the previous round, the utility of party j is greater than that of party i if the last object allocated to party i is removed. □

However, the property of EF1 does not determine any bound on the difference in utilities for all parties. In our allocation, since a correlated equilibrium is maintained while choosing the objects, we can also prove an absolute bound on the utility difference between parties.

Lemma 14. *(Bounded Envy) In Algorithm 1, the utility for each party is at least half the utility of all other parties.*

Loop	U_{curr}	s	N	U after update
1	[8, 6, 5, 3, 1]	0	[1, 0, 0, 0, 0]	[8, 6, 5, 3, 1]
2	[4, 6, 5, 3, 1]	1	[1, 1, 0, 0, 0]	[8, 6, 5, 3, 1]
3	[4, 3, 5, 3, 1]	2	[1, 1, 1, 0, 0]	[8, 6, 5, 3, 1]
4	[4, 3, 2.5, 3, 1]	0	[2, 1, 1, 0, 0]	[8, 6, 5, 3, 1]
5	[2.66, 3, 2.5, 3, 1]	1	[2, 2, 1, 0, 0]	[8, 6, 5, 3, 1]
6	[2.66, 1.5, 2.5, 3, 1]	3	[2, 2, 1, 1, 0]	[8, 6, 5, 3, 1]

Round	Description	Variable State
1	Assign $A[0]$ once, $A[1]$ twice	L = [[0, 0, 1, 0, 0], [1, 0, 0, 0, 0], [1, 0, 0, 0, 0]]
2	Assign $A[2]$ twice, $A[3]$ once	L = [[0, 1, 1, 0, 0], [1, 1, 0, 0, 0], [1, 0, 0, 1, 0]]

Proof. We maintain the following invariant while selecting objects in algorithm 1, except when $n_j = n$.

$$\frac{u_i}{n_i} \geq \frac{u_j}{n_j + 1}$$

If $n_j = n$, then the object would be allocated to all n parties since all parties are allocated an object only once. Let's say some party that picks object o_i in the last round envies the party that picked o_j in the first round.

$$\frac{u_{o_i}}{n_{o_i}} \geq \frac{u_{o_j}}{n_{o_j} + 1}$$

$$\frac{u_{o_j}}{n_{o_j}} \geq \frac{u_{o_i}}{n_{o_i}}$$

In the proof for Lemma 13, we have that utility for a party with object o_j is lower than the utility for a party with object o_i if both o_j and o_i are removed.

$$\mathbb{U}(L_i \setminus \{o_i\}) \geq \mathbb{U}(L_j \setminus \{o_j\})$$

The difference for the party with object o_i who envies the party with object o_j is given by,

$$\begin{aligned}
&\mathbb{U}(L_j) - \mathbb{U}(L_i) \\
&\mathbb{U}(L_j \setminus \{o_j\}) + \frac{u_{o_j}}{n_{o_j}} - \mathbb{U}(L_i \setminus \{o_i\}) - \frac{u_{o_i}}{n_{o_i}} \\
&\leq \mathbb{U}(L_j \setminus \{o_j\}) + \frac{u_{o_j}}{n_{o_j}} - \mathbb{U}(L_i \setminus \{o_i\}) - \frac{u_{o_j}}{n_{o_j} + 1} \\
&\leq \frac{u_{o_j}}{n_{o_j}} - \frac{u_{o_j}}{n_{o_j} + 1} \leq \frac{u_{o_j}}{n_{o_j} \cdot (n_{o_j} + 1)} \leq \frac{u_{o_j}}{n_{o_j} \cdot 2} \leq \frac{\mathbb{U}(L_j)}{2}
\end{aligned}$$

□

Now that we have bounded the envy, can we do better? In another definition of Envy Freeness in [28], we have

Lemma 15. *(EFx-Fairness) The allocation achieved in Algorithm 1 is EFx Fair.*

Proof. Consider party j envies party i, i.e. $\mathbb{U}(L_i) \geq \mathbb{U}(L_j)$. Let u_p^r be the utility from r^{th} object allocated to party $p \in \{i, j\}$. We know that

$$\frac{u_i^r}{n_i^r} \geq \frac{u_j^r}{n_j^r} \ \forall r$$

$$\frac{u_p^{r-1}}{n_p^{r-1}} \geq \frac{u_{p'}^r}{n_{p'}^r} \ \forall p, p' \in \{i, j\}, \forall r$$

Consider all objects allocated to party i in rounds $2, \ldots, k-1$.

$$\begin{aligned}
&\frac{u_j^{r-1}}{n_j^{r-1}} \geq \frac{u_i^r}{n_i^r} \ \forall r \in \{2, \ldots, k-1\} \\
&\sum_{r \in \{1, \ldots, k-2\}} \frac{u_j^r}{n_j^r} \geq \sum_{r \in \{2, \ldots, k-1\}} \frac{u_i^r}{n_i^r} \qquad (11)
\end{aligned}$$

Further,

$$\begin{aligned}
\frac{u_i^1}{n_i^1} &\geq \frac{u_j^k}{n_j^k} \\
\frac{u_j^k}{n_j^k} \geq \frac{u_i^1}{n_i^1 + 1} &\geq \frac{u_i^1}{2n_i^1} \\
\frac{u_j^{k-1}}{n_j^{k-1}} \geq \frac{u_j^k}{n_j^k} &\geq \frac{u_i^1}{2n_i^1} \\
\frac{u_j^{k-1}}{n_j^{k-1}} + \frac{u_j^k}{n_j^k} &\geq \frac{u_i^1}{n_i^1} \qquad (12)
\end{aligned}$$

Adding (11) and (12),

$$\sum_{r \in \{1, \ldots, k\}} \frac{u_j^r}{n_j^r} \geq \sum_{r \in \{1, \ldots, k-1\}} \frac{u_i^r}{n_i^r} \qquad (13)$$

Which implies $\mathbb{U}(L_j) \geq \mathbb{U}(L_i \setminus o_i^k)$ where o_i^k is the lowest utility item in L_i, and thus proving the algorithm satisfies EFx Fairness. □

Thus, we achieve the following fairness definition.

Theorem 4. *(Fairness) Algorithm 1 achieves an EFx fair allocation with the utility of each party being at least half of any other party participating in the allocation.*

Allocation	Bribery Budgets							
Fee tx	15	14	13	11	6	5	4	3,2,1,1,1,1,1,1
A0	8	6.5	5.5	3.5	0	0	0	0
A1	46.583	32.083	28.833	15.25	2.333	1.333	0.333	0
A2	46.25	31.75	28.5	22.25	2	1	0	0
A3	44.75	40.75	27	20.75	1	0	0	0
A4	49	44	39	0	0	0	0	0

TABLE 3: Censorship Resistance provided by each allocation.

The proof follows from Lemma 14 and Lemma 15.